CS4552: Network Design and Programming

Geoffrey Xie

Course Scope

Project 1 (35% of grade) – Due Week 3

- **Build a local area network (LAN) consisting of several PCs hosting Windows 2000 Server and a single Windows 2k/XP Workstation**
  - create and maintain user accounts and directories
  - create a networked file system and install authorization tools (ACL)
  - install HTTP, FTP, Web, and print servers
  - install DNS, DHCP, and VPN services
  - enabling routing protocols
  - (connect to the Internet)
  - Demonstrate all functionalities
- **Build a LAN of Linux workstations using the same hardware**

Course Scope (cont’d)

Project 2 (15% of grade) – Due Week 5

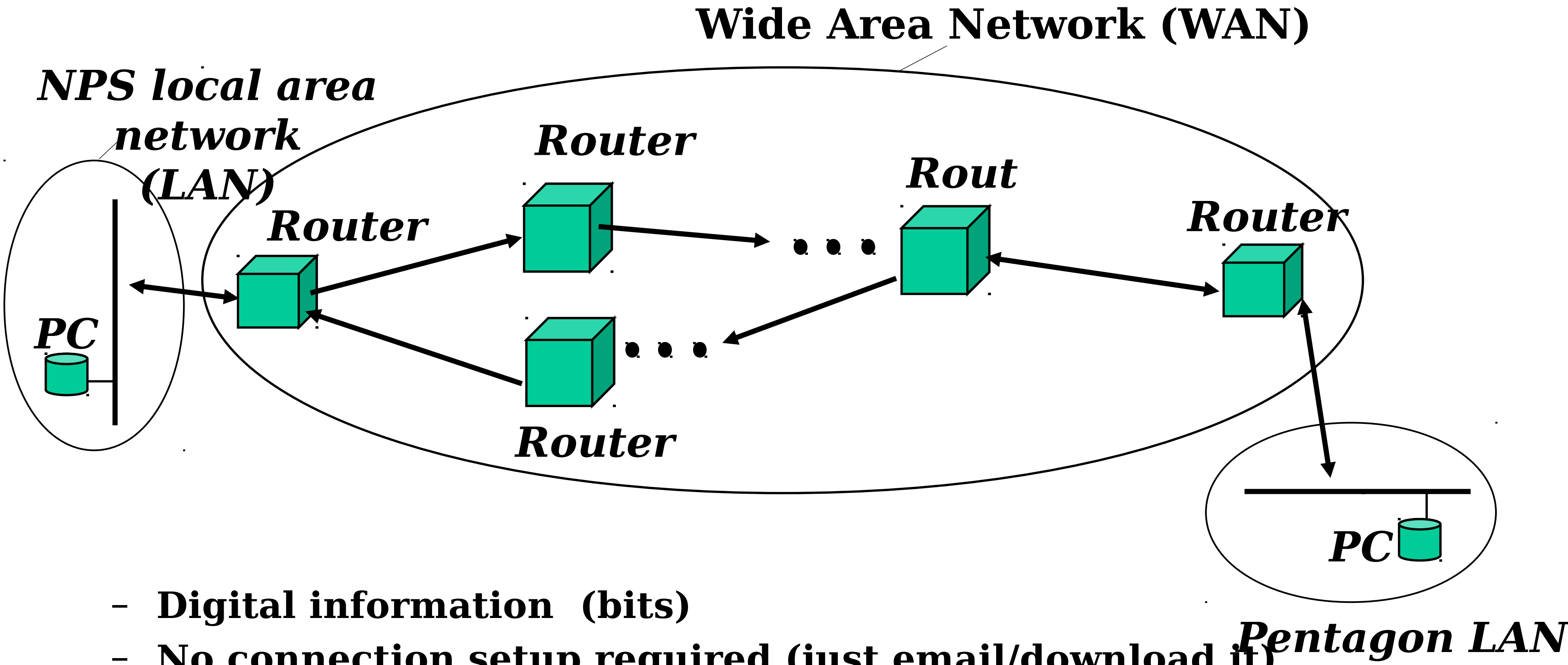
- **Implement Autonomous System (AS) Routing**
    - Configure a collection of PCs hosting either Linux or Windows 2K Server as two Autonomous Systems composed of three routers each
    - Implement OSPF intra-AS routing within each AS
    - Implement BGP inter-AS routing between the two configured AS
    - Implement BGP between groups

Course Scope (cont’d)

Project 3 (50% of grade) – Due Week 11

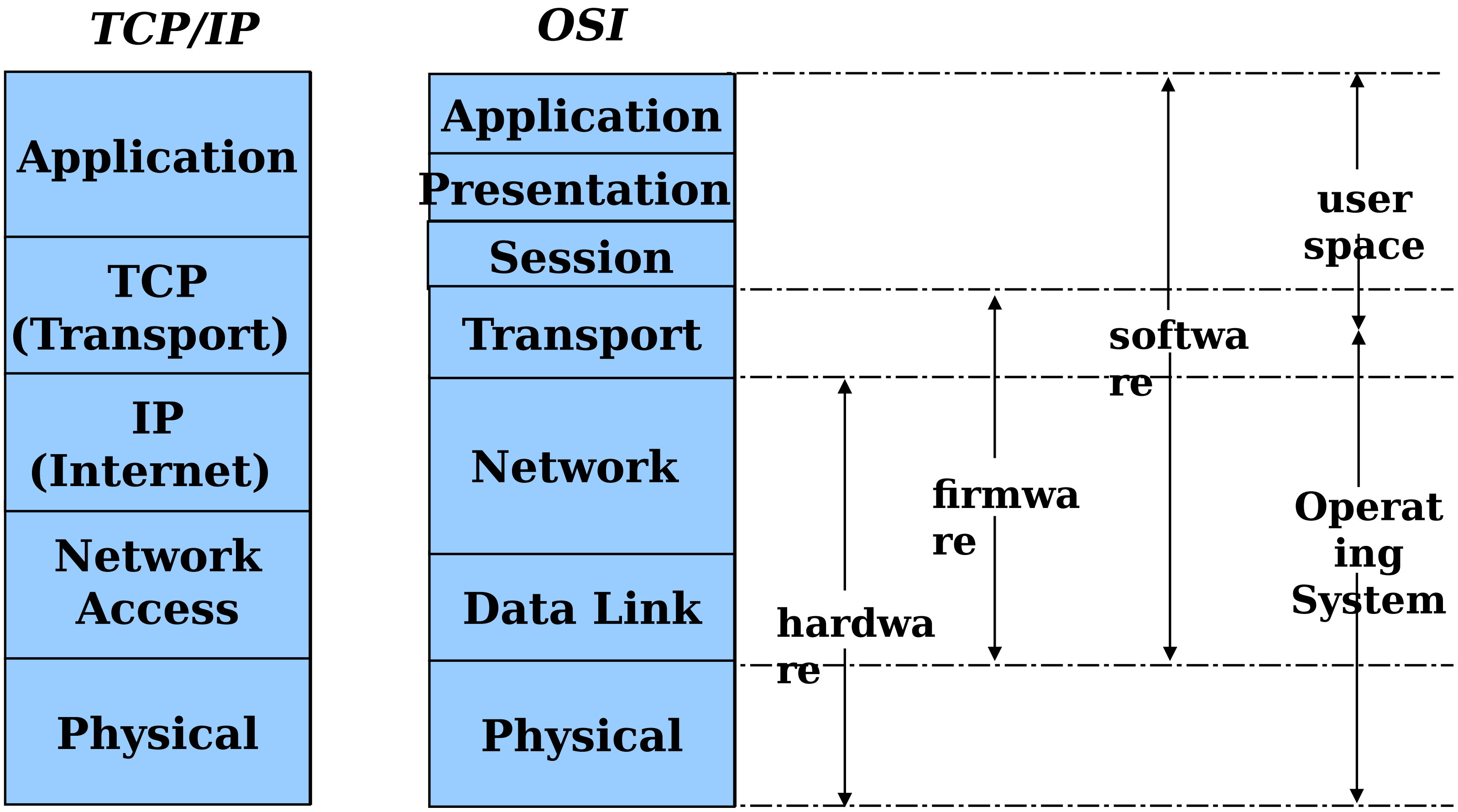
- **Investigate emerging or promising networking technologies**
- **Topics to be announced during Week 4, but in general affect:**
  - Service to mobile forces
  - Multi-service networks
  - Distributed applications
  - Media access
  - Protocol analysis and performance
  - Security….

Review of Computer Networking Basics



- Digital information (bits)
- No connection setup required (just email/download it)
- Simplex channel (each message routed separately)
  - Store and forward model for routers
- Best effort service (timely delivery of messages *most of the time*)
  - No hard guarantee on performance

Layered Model of Network Architecture



TCP/IP

Application

TCP (Transport)

IP (Internet)

Network Access

Physical

OSI

Application

Presentation

Session

Transport

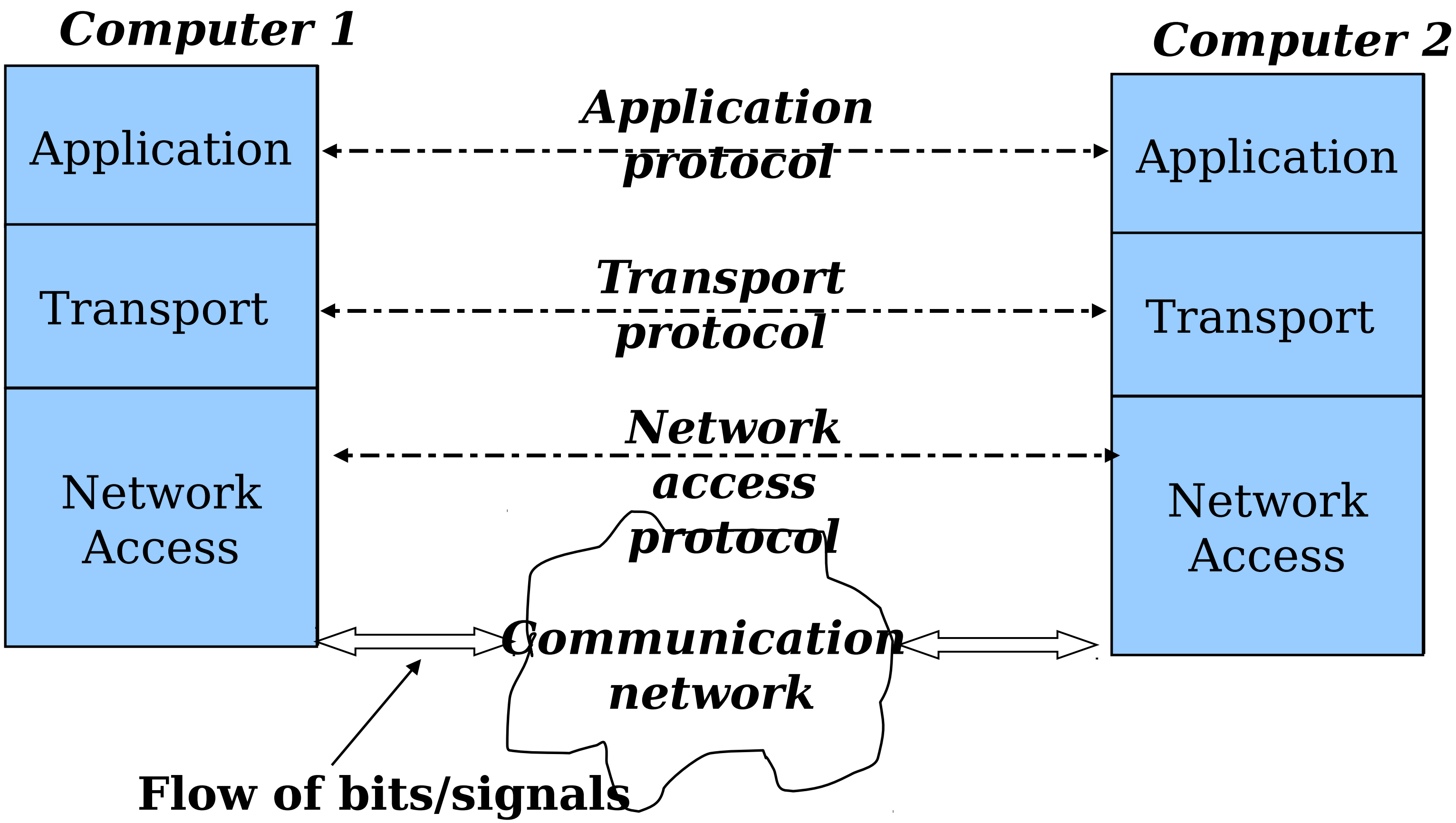
Network

Data Link

Physical

Each layer provides a set of services – a set of function calls – an interface – to its immediate upper layer.

Peer-Level Communication



Computer 1

Application

Transport

Network Access

Computer 2

Application

Transport

Network Access

Application protocol

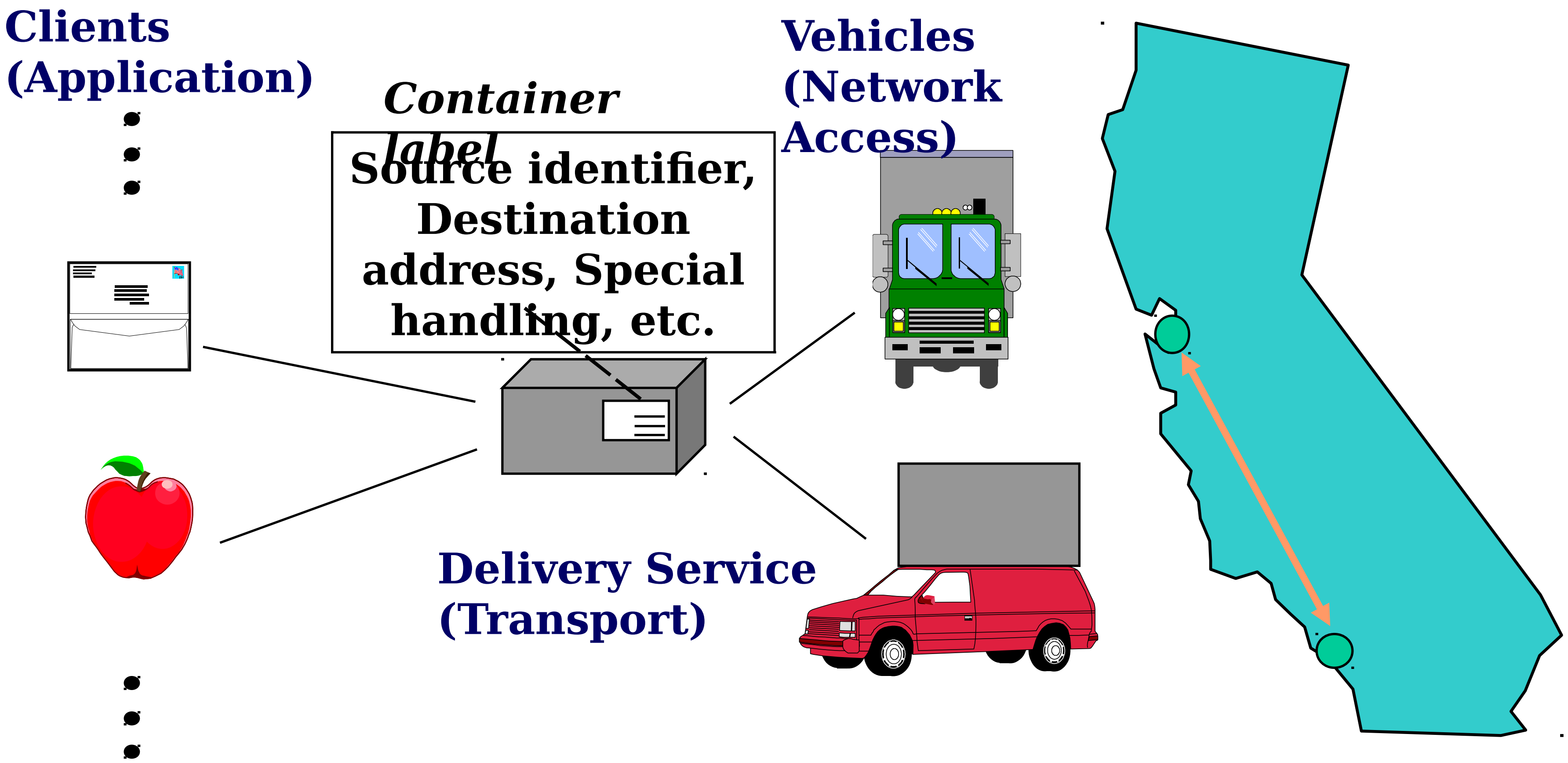
Transport protocol

Network access protocol

Communication network

Flow of bits/signals

Delivery Service Example



Clients (Application)

Container label

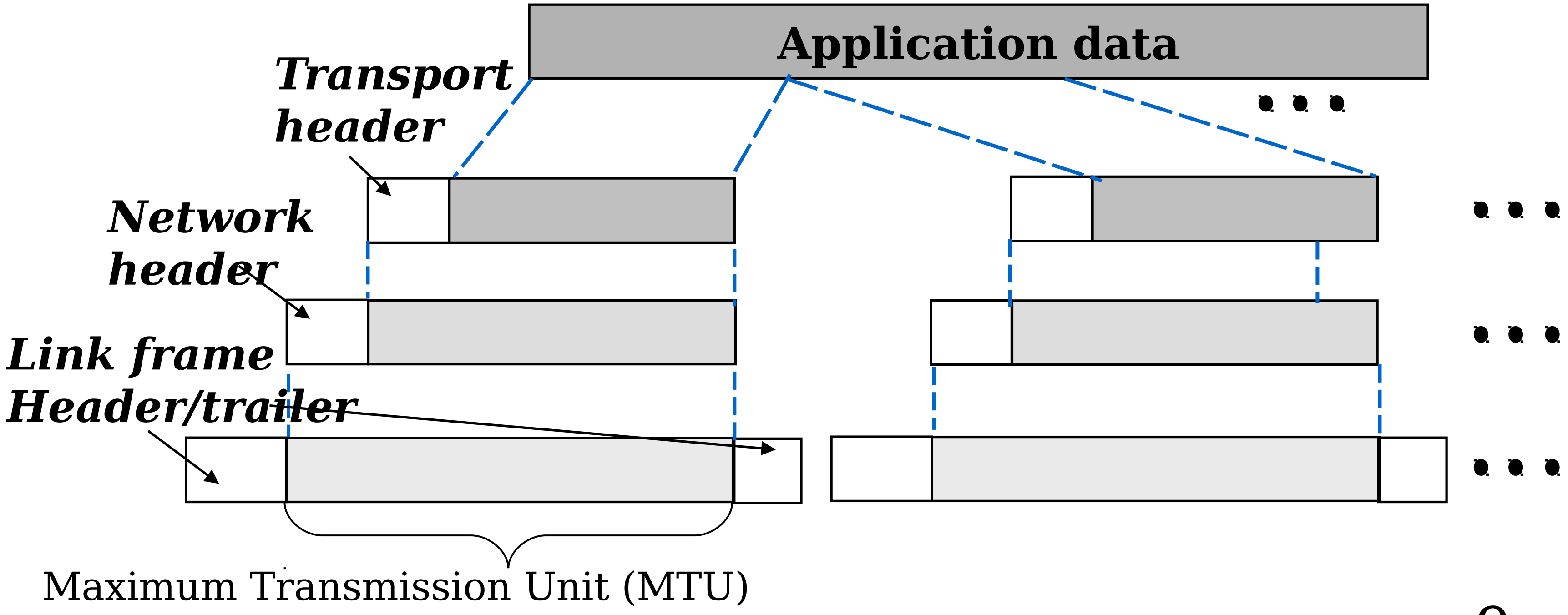
Source identifier, Destination address, Special handling, etc.

Vehicles (Network Access)

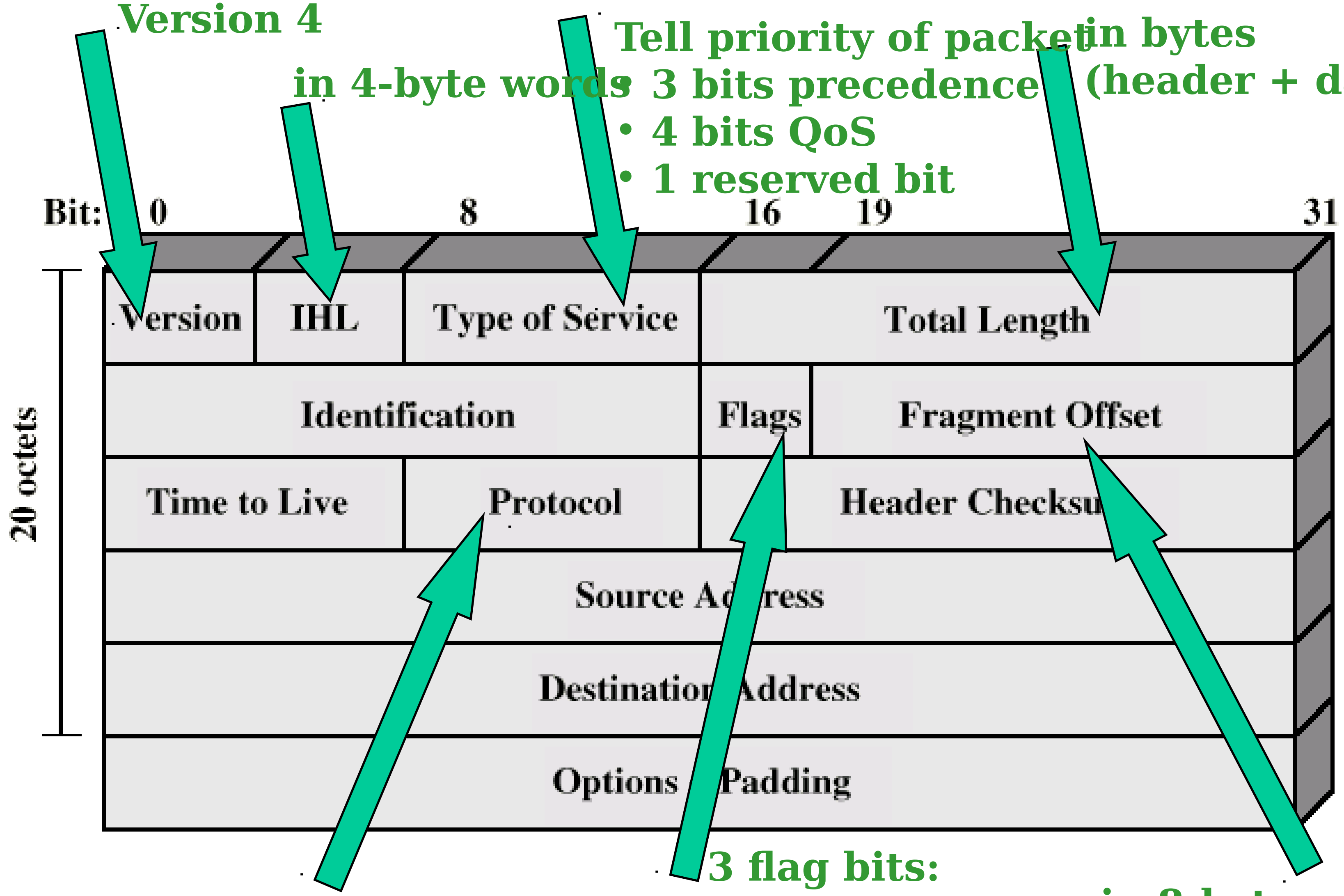
Delivery Service (Transport)

Protocol Data Units

- **Data encapsulation to facilitate layered protocol architecture**
  - PDU functions like truck container
- **Protocol header**
  - addressing: identification of receiving entity
  - control information: sequencing, special routing request



Internet Protocol (v4) Header



Version 4

in 4-byte words

Tell priority of packet
- 3 bits precedence
- 4 bits QoS
- 1 reserved bit

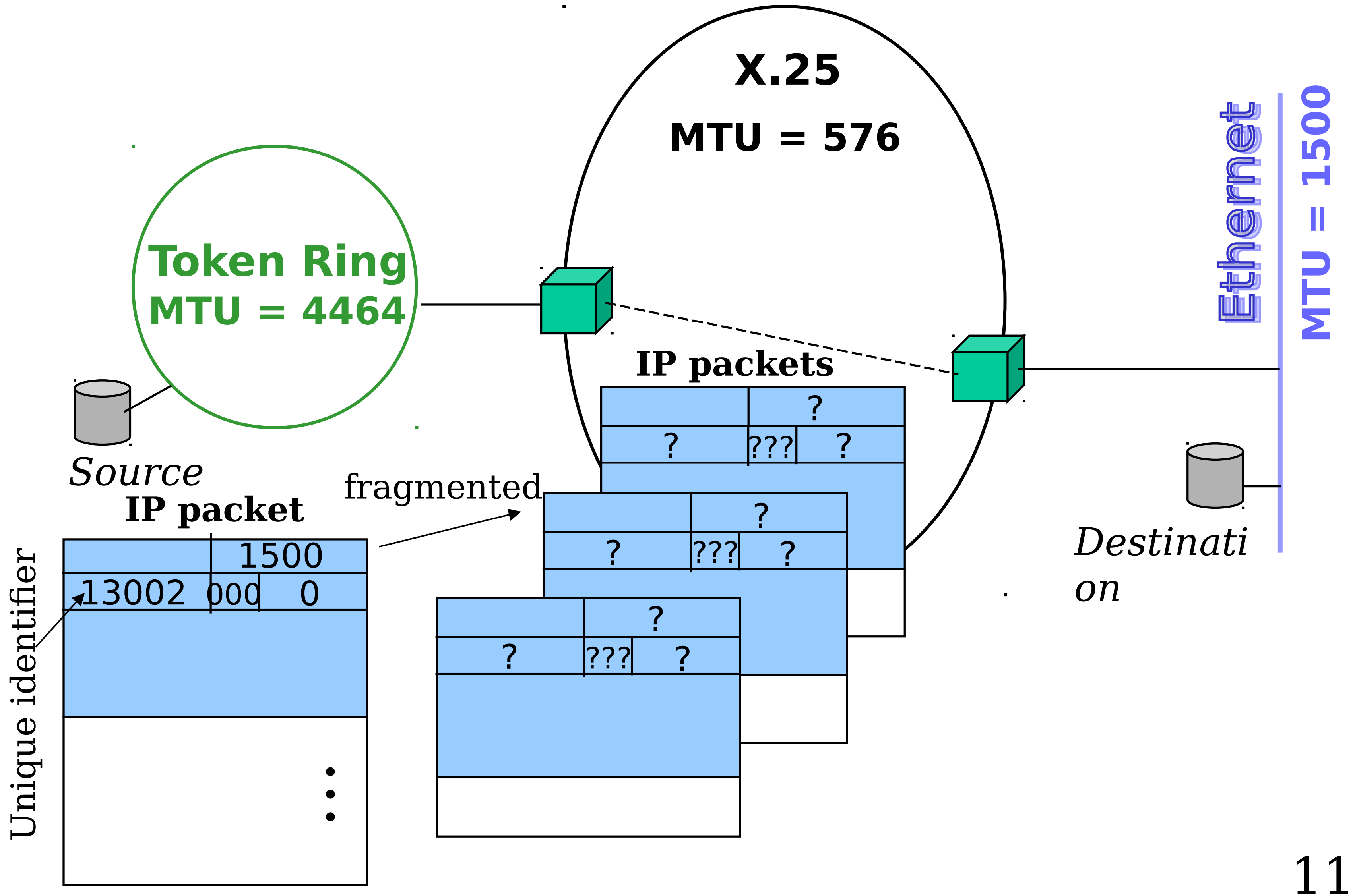
in bytes
(header + data)

Tell type of payload
(6 – TCP; 17 – UDP; etc)

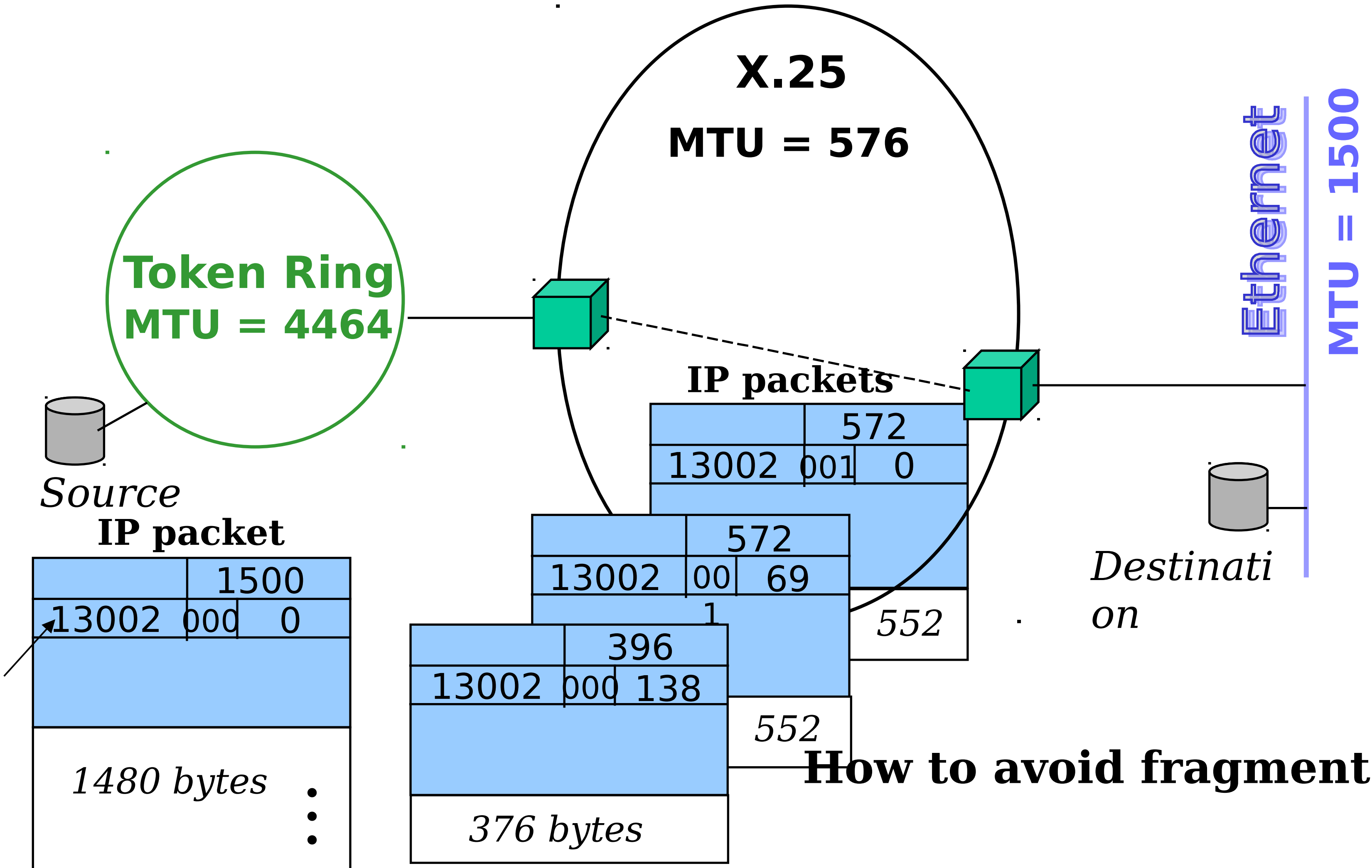
3 flag bits:
| unused | DF |
MF |

in 8-byte
words
(from start of

IP Fragmentation and Reassembly



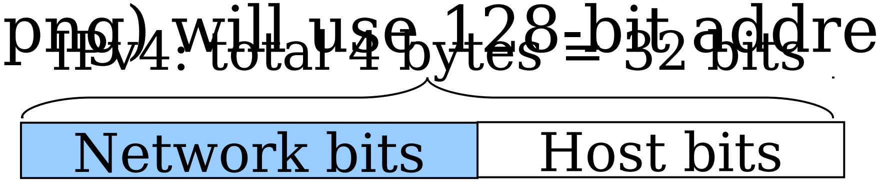
IP Fragmentation and Reassembly (solution)



How to avoid fragmentation?

Internet Protocol Addresses

- **IPv4 address hierarchy**
  - network classes:
    - **A** (8 network bits: 0…), **B** (16 net bits: 10…), **C** (24 net bits: 110…)
    - **D** for multicast (1110…) and **E** reserved for future use (11110…)
  - facilitate hierarchical routing
  - IPv6 (Ipng) will use 128-bit addresses



IPv4: total 4 bytes = 32 bits

Network bits

Host bits

Subnets

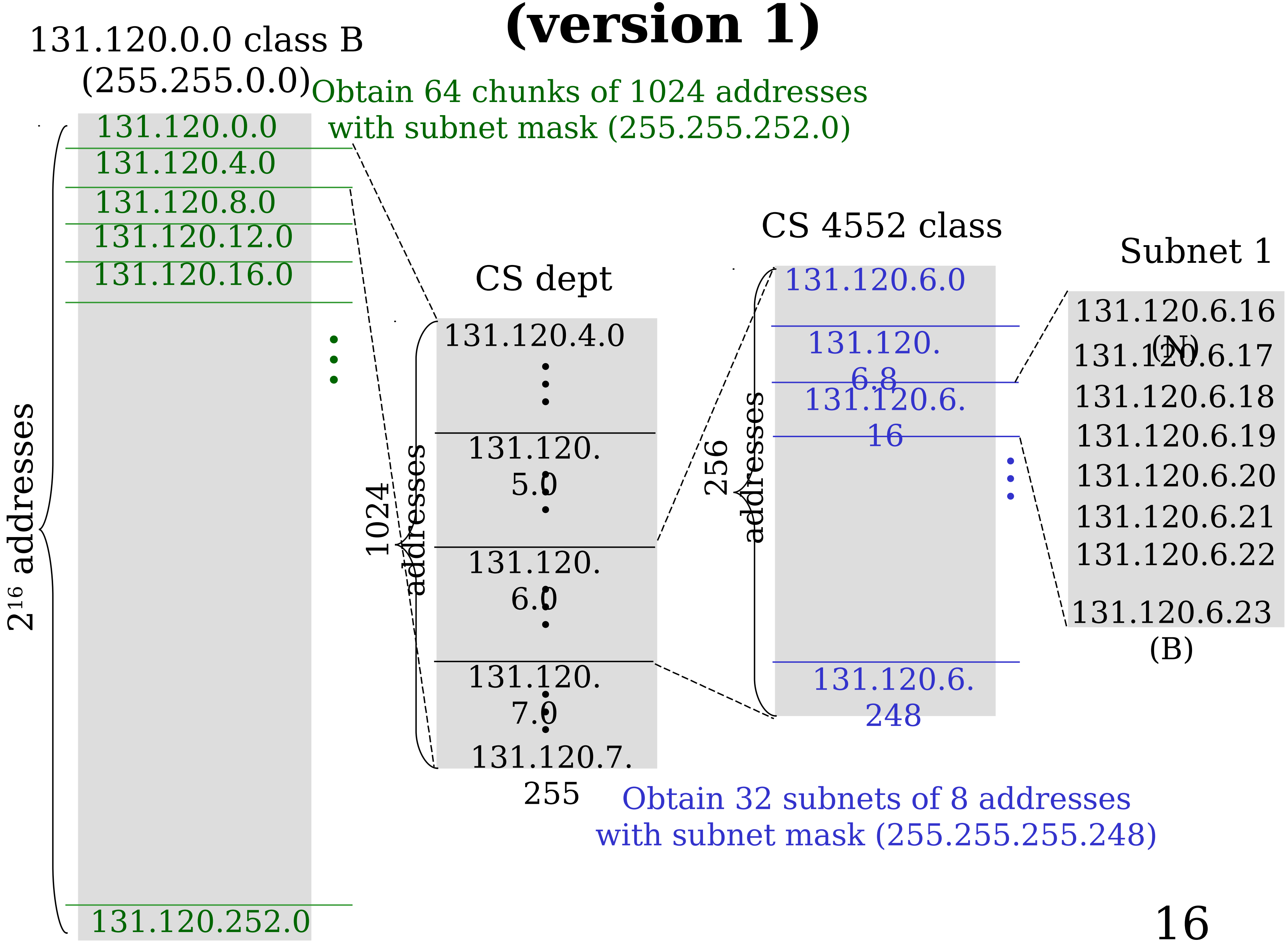
- **User defined address hierarchy within a class A, B, or C network**
  - more network bits and fewer host bits than normal
    - Example: How many more network bits are required if we want to partition a class C network (e.g., 194.120.8.0) into 9 subnets of the same size?
  - subnet mask: "1" for all net bits and "0" for all host bits
    - All hosts on one subnet must use the same subnet mask. Why?
    - What is the network mask in the above subnetting example? How many IP addresses are available for hosts in each subnet?
    - Two representations: e.g., 255.255.255.0 ←→ /24

Functionality of Subnet Mask

- **Help host/gateway determine if a destination IP address is inside the same LAN**
  - Yes if (ownIP & NetMask == destIP & NetMask)
    - Host/gateway then consults the Address Resolution Protocol (ARP) server to find the MAC address of destination
  - Otherwise,
    - host: forward packet to the gateway
    - gateway: route packet based o
      - "longest match first"
      - Use default route upon no ma

Net/Netmask	**Next Hop**
121.5.3.0/24	**eth01**
131.120.0.0/16	**13.120.4.1**
......	**...**

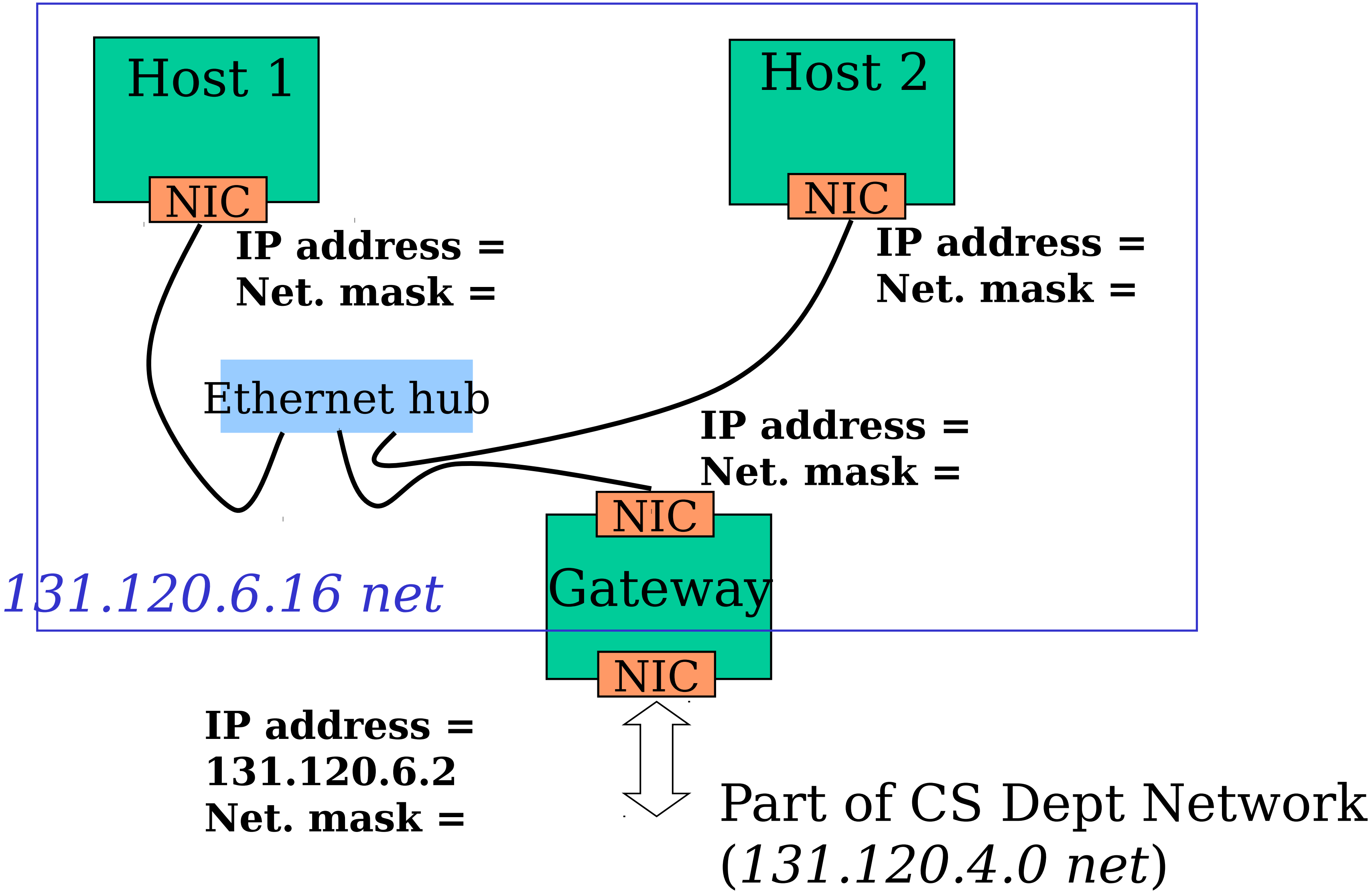
Address Maps for CS4552 Lab (version 1)



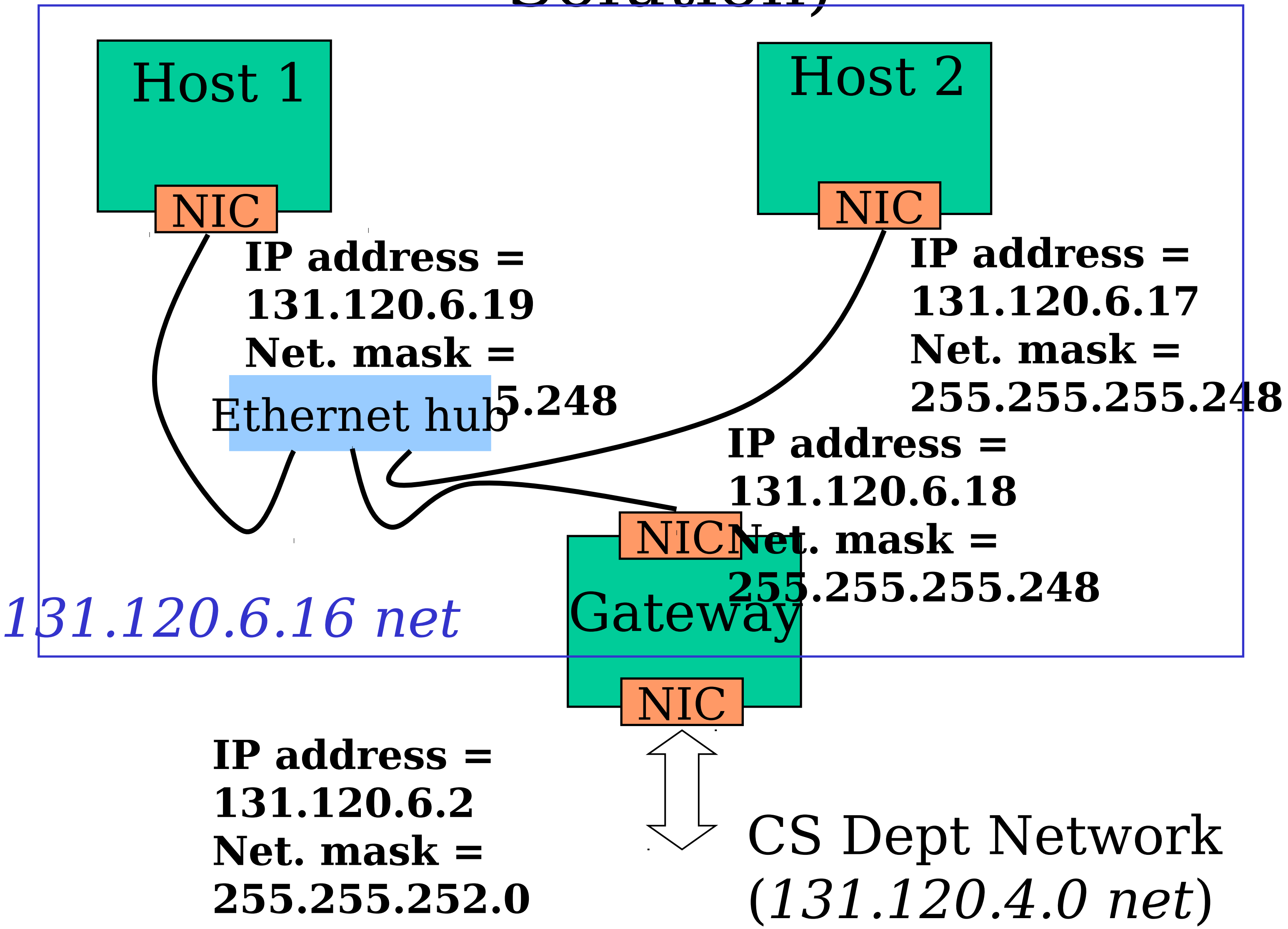
Obtain 64 chunks of 1024 addresses with subnet mask (255.255.252.0)

Obtain 32 subnets of 8 addresses with subnet mask (255.255.255.248)

A 3-Node Subnet (version 1)



A 3-Node Subnet (Version 1 Solution)



Host 1

NIC

IP address =
131.120.6.19
Net. mask =
5.248

Ethernet hub

Host 2

NIC

IP address =
131.120.6.17
Net. mask =
255.255.255.248

NIC

IP address =
131.120.6.18
Net. mask =
255.255.255.248

Gateway

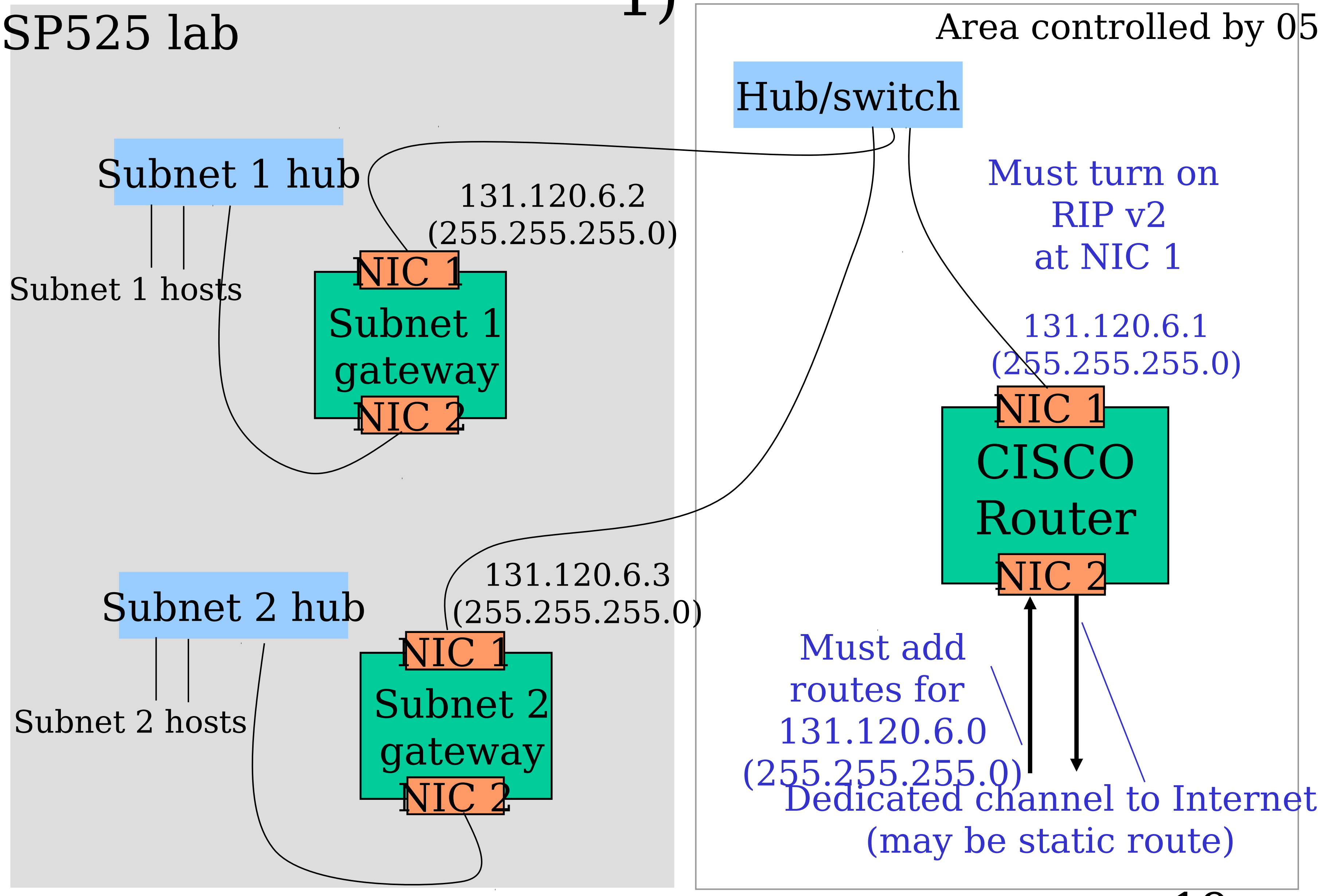
131.120.6.16 net

NIC

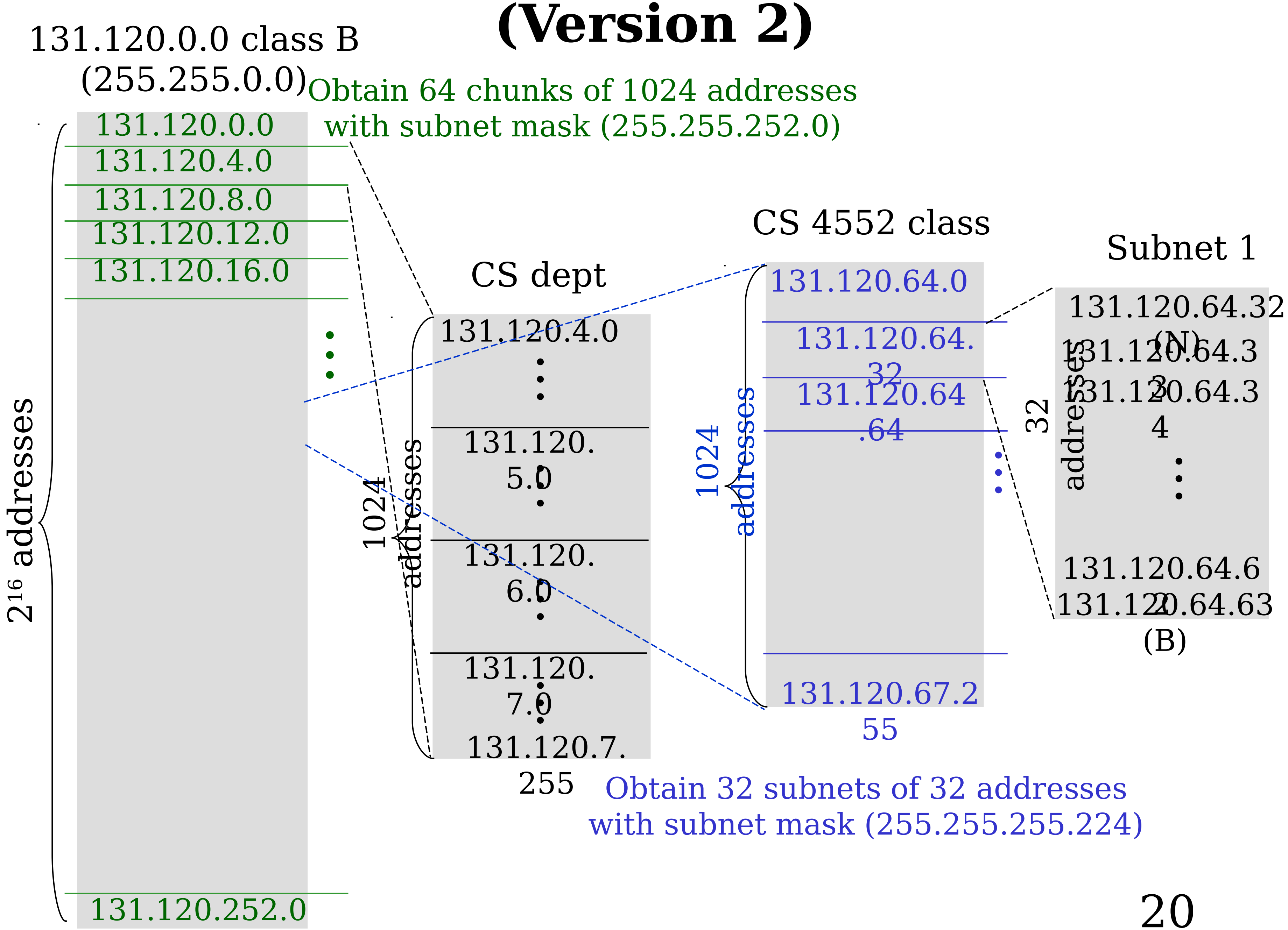
IP address =
131.120.6.2
Net. mask =
255.255.252.0

CS Dept Network
(*131.120.4.0 net*)

Outside Connect via Router (Version 1)



Address Maps for CS4552 Lab (Version 2)

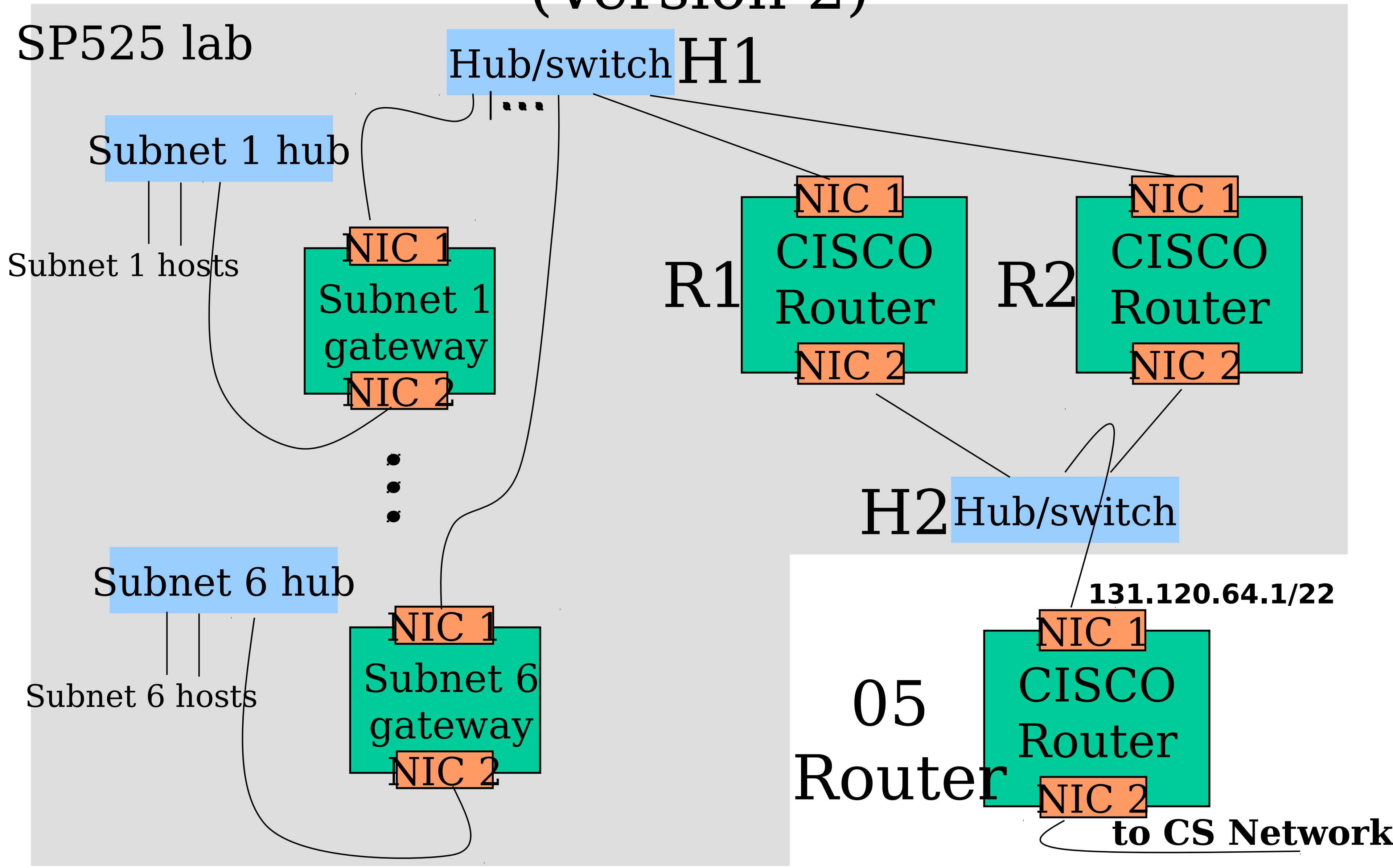


131.120.0.0 class B (255.255.0.0)

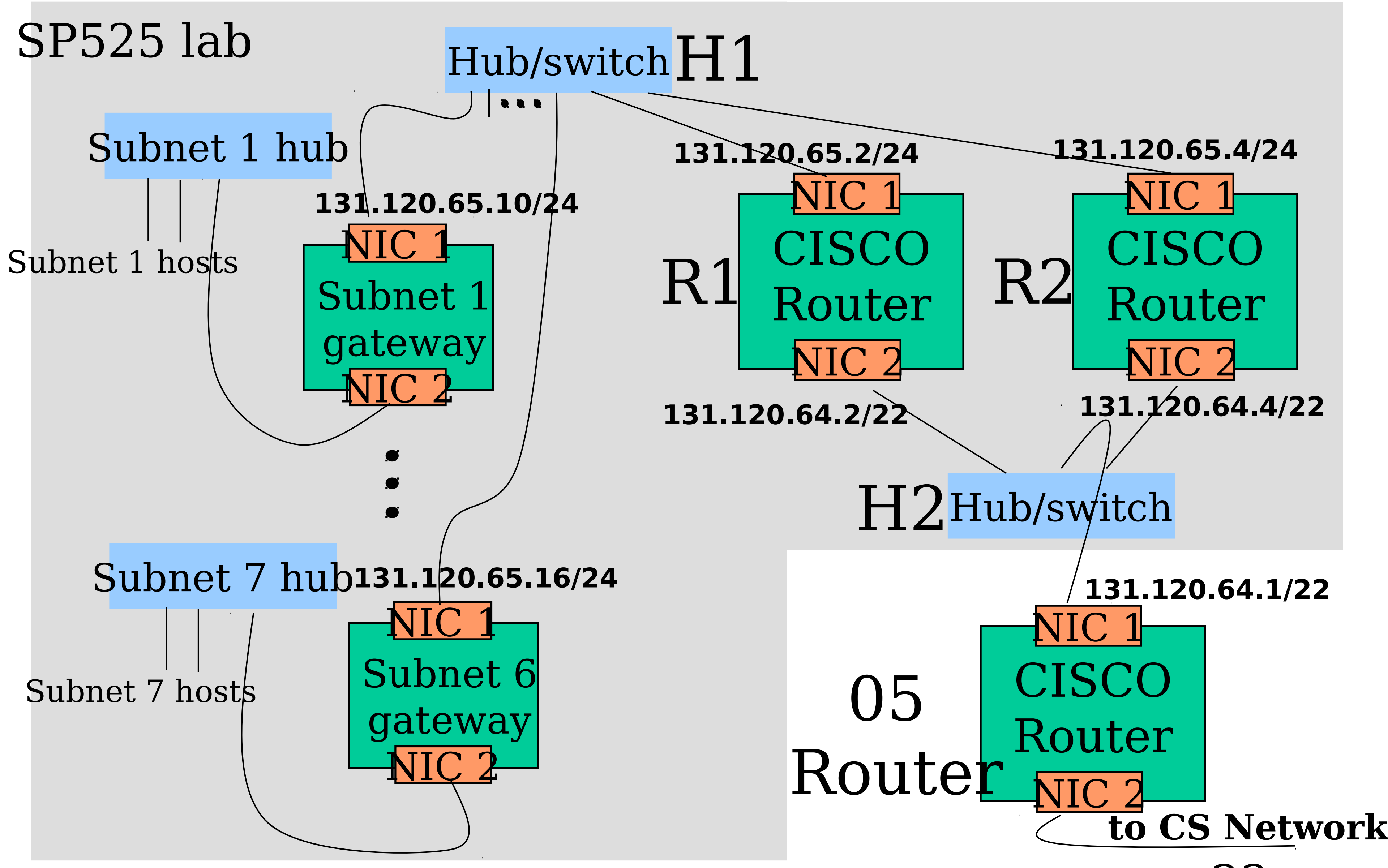
Obtain 64 chunks of 1024 addresses with subnet mask (255.255.252.0)

Obtain 32 subnets of 32 addresses with subnet mask (255.255.255.224)

Outside Connect via Three Routers (Version 2)



Outside Connect (Version 2 Solution)



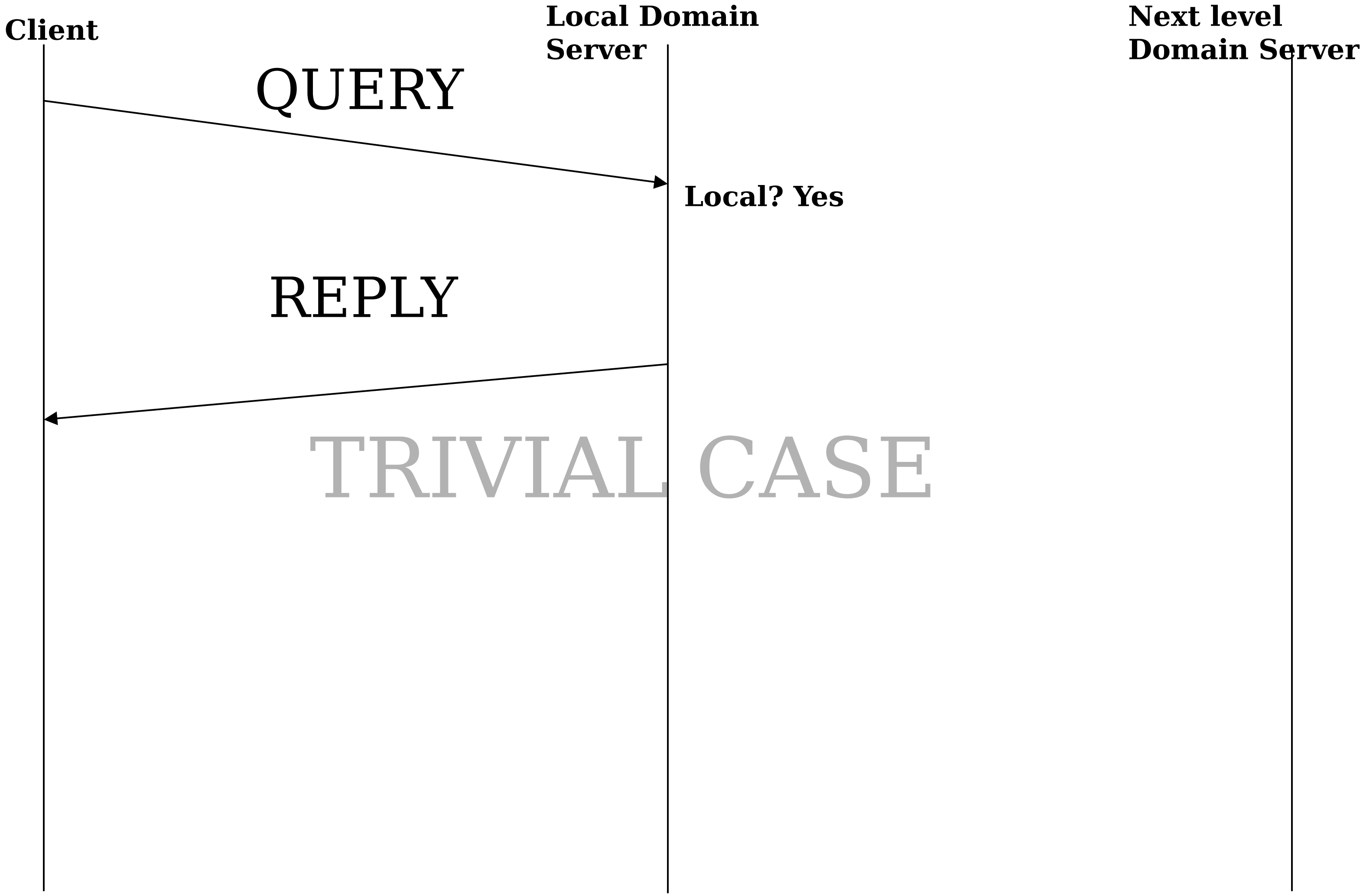
Domain and Gateway

- **Domain: address hierarchy for computer hosts in Internet**
  - just like street address hierarchy for Post Office
  - email-address format: <user>@<domain>
    - e.g., xie@cs.nps.navy.mil
  - host name format: <host>.<domain>
    - <u>nickname for IP address</u>
    - e.g., taurus.cs.nps.navy.mil ←→ 131.120.10.2
- **Gateway**
  - routing packets in and out of a domain
  - at least one per LAN

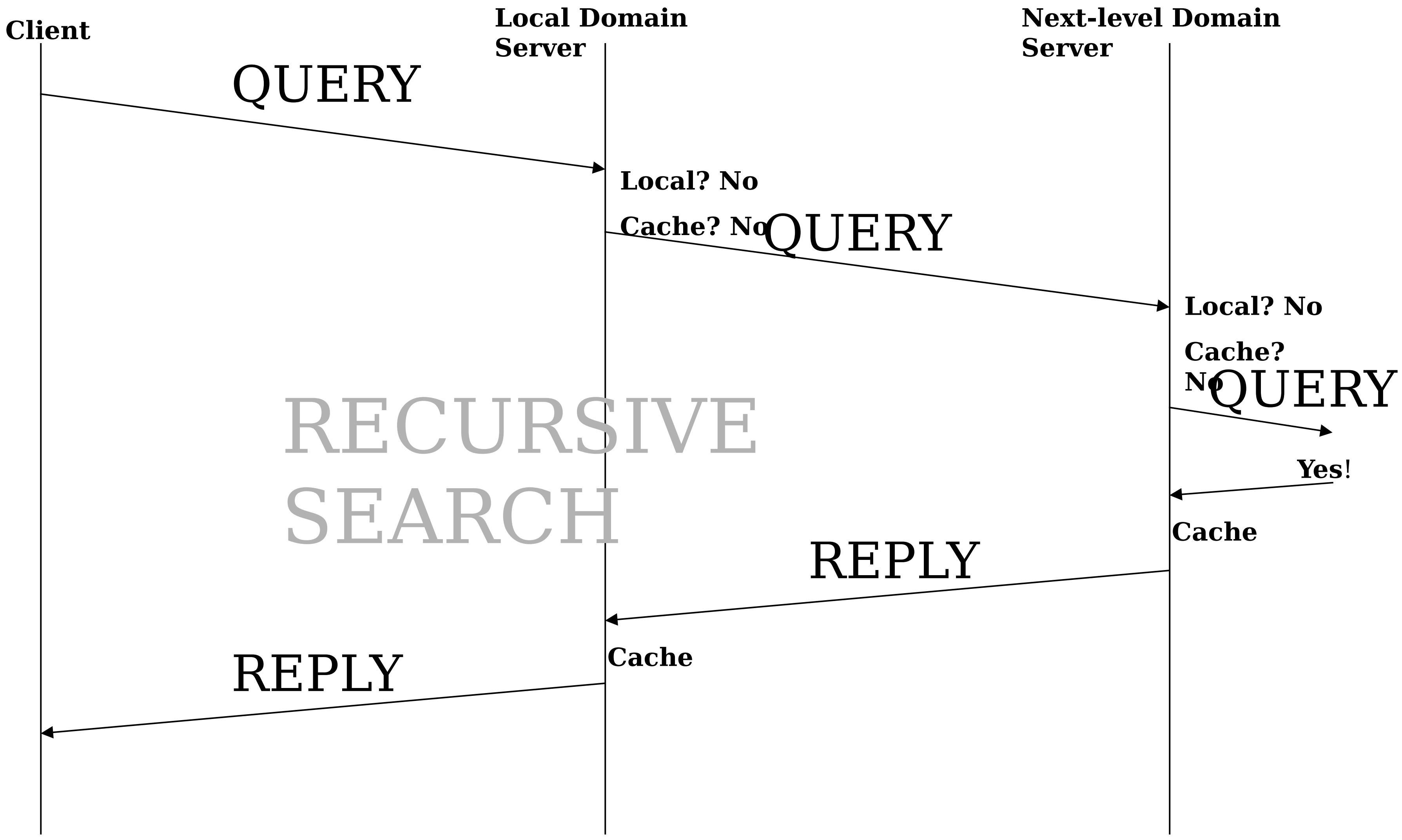
Domain Name Service (DNS)

- **Needed for IP address resolution based on host name (RFC 1035)**
  - Mainly two types of Resource Record (RR):
    - **A** record: mapping Fully Qualified Domain Name (FQDN) to IP address
    - **PTR** record: mapping an IP address to a FQDN
  - loopback address for local host: 127.0.0.1
- **Flat “hosts” file does not work because of Internet’s large size and its dynamic nature**
  - hosts are added, moved and removed constantly
- **Carried out by a hierarchy of servers**
  - each server maintains a small number of entries
  - caching may be used to improve performance
  - DNS messages are communicated via UDP (or TCP) port 53

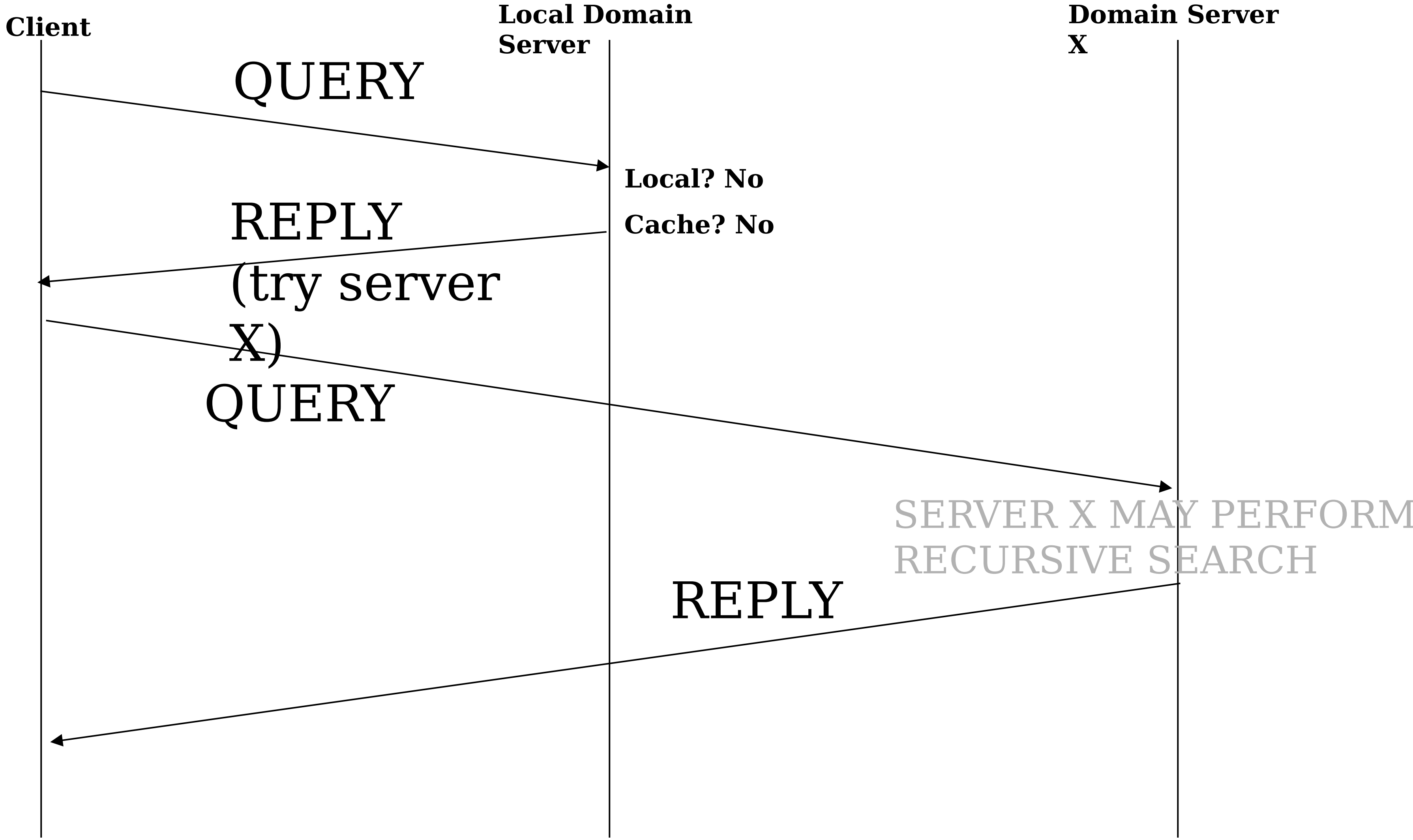
DNS Client/Server Interactions



DNS Recursive Search



DNS Server Delegation



Client

Local Domain Server

Domain Server X

QUERY

Local? No

Cache? No

REPLY (try server X)

QUERY

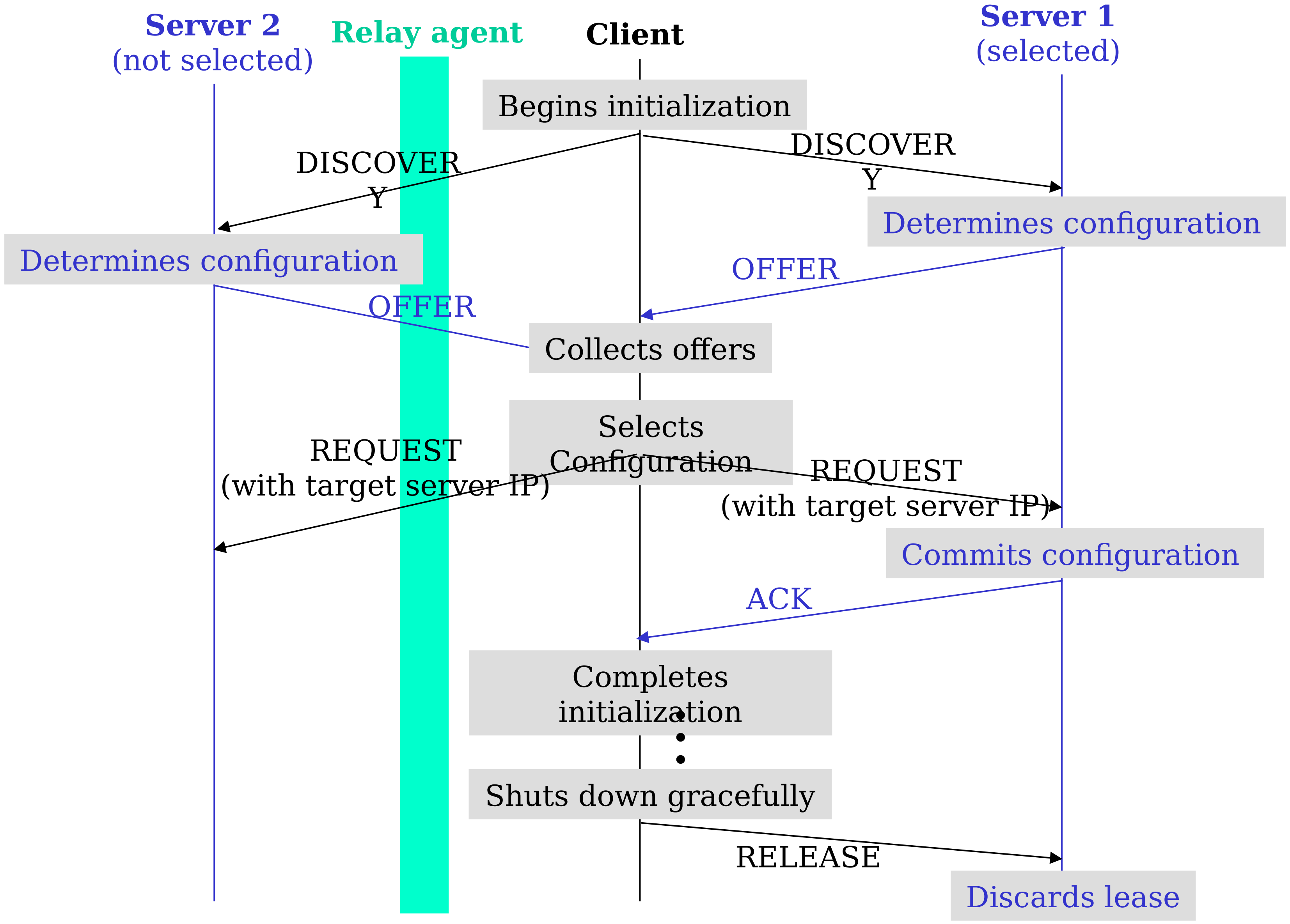
SERVER X MAY PERFORM RECURSIVE SEARCH

REPLY

Dynamic Host Configuration Protocol (DHCP)

- **Needed for dynamic configuration of network hosts**
  - extension of BOOTP (RFC951)
  - DHCP (BOOTP) messages are transported via UDP port 67 and 68
  - may automatically notify DNS of new address allocations
- **Support three types of IP address allocation**
  - Dynamic: address allocated to a host for a finite lease time
  - Automatic: address is allocated to a host with infinite lease
  - Static: address for a host is chosen by administrator
- **DHCP messages may be relayed across multiple subnets**
  - DHCP messages are broadcasted within a subnet; all messages of one session carry a unique integer identifier randomly generated by client
  - Unicast between relay agents and DHCP server

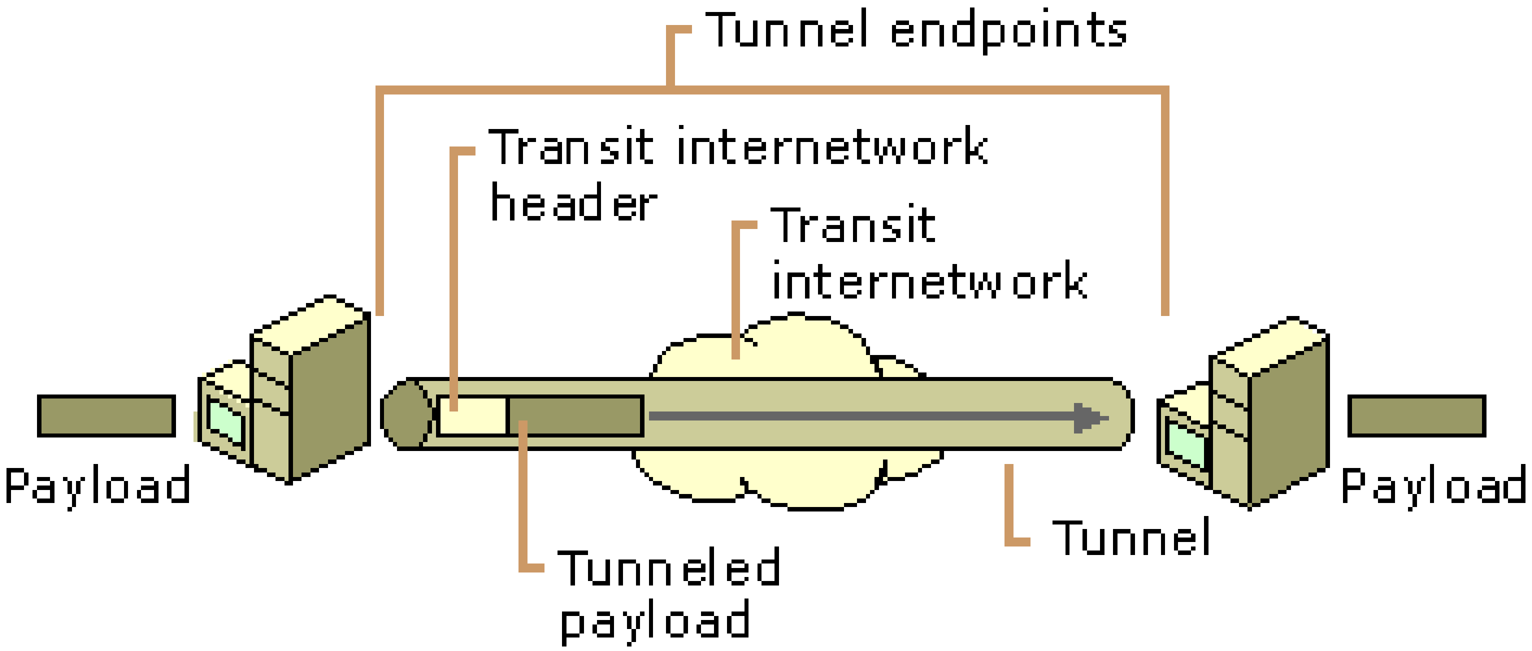
DHCP Client/Server Interactions



Virtual Private Network (VPN)

- **Private networks connected via logical tunnels through public networks**
  - Access control
  - Source authentication
  - Data integrity
  - Encryption for data confidentiality
- **IPSec (IP Security)**
  - Tunneled at IP layer, i.e., packet encapsulated in another IP packet
- **PPTP (Point-to-Point Tunneling Protocol) / L2TP (Layer 2 Tunneling)**
  - Tunneled at link layer

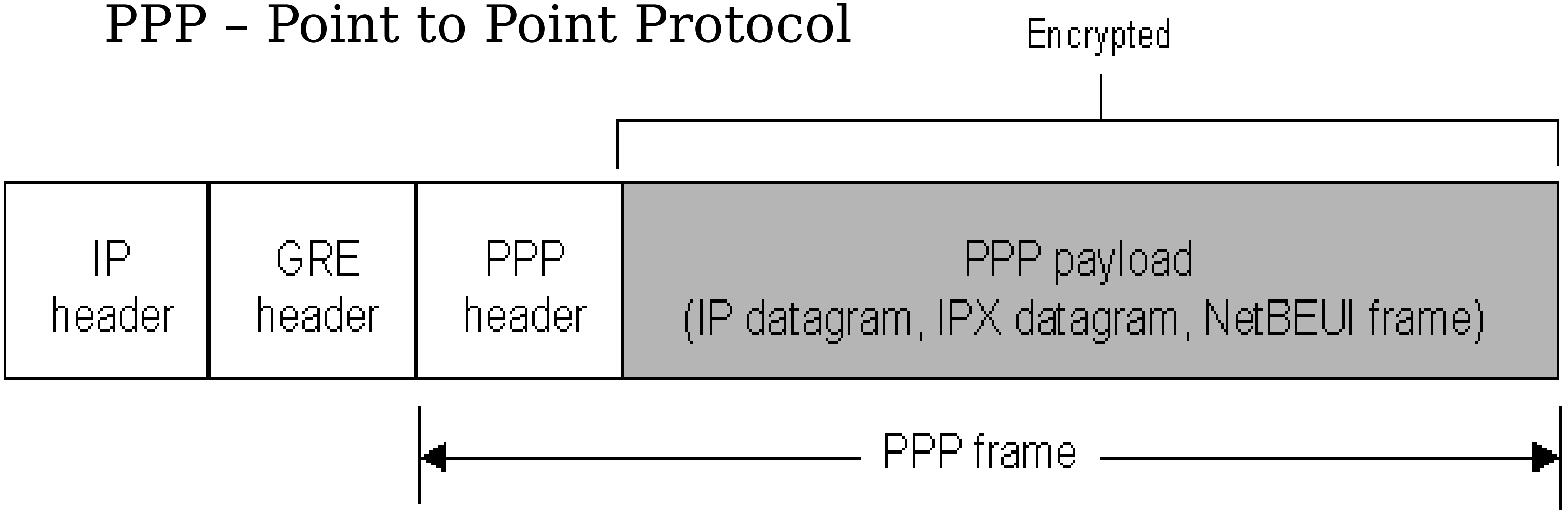
Tunneling



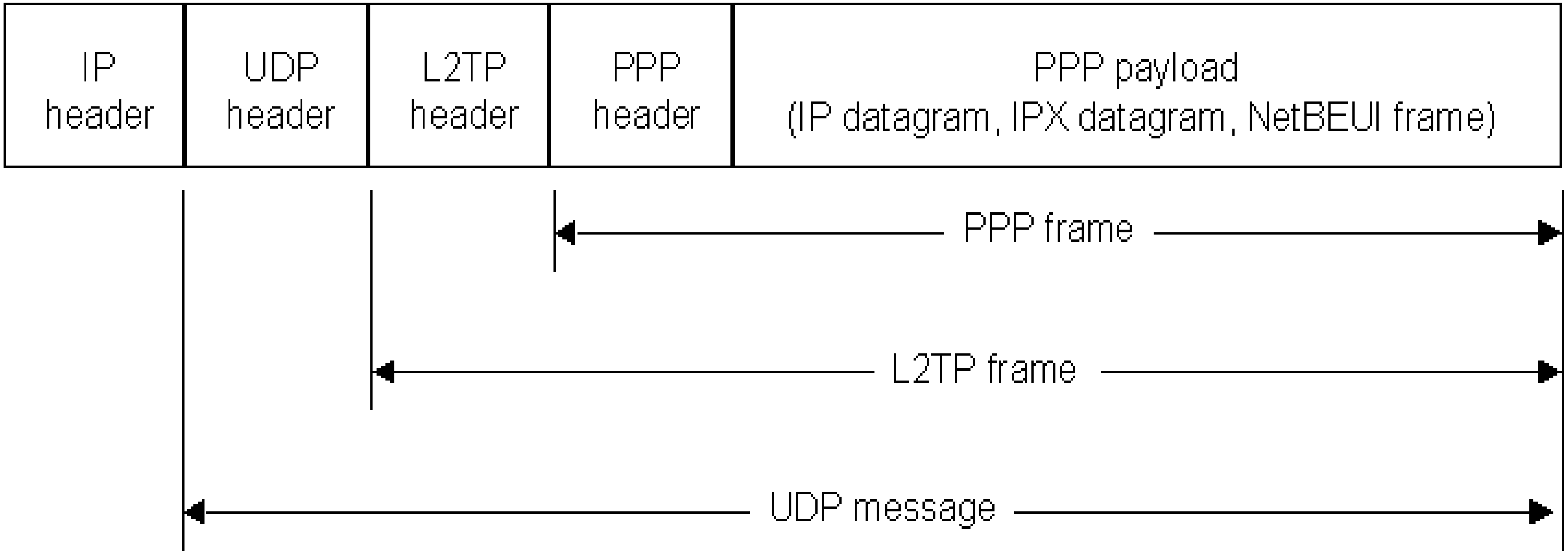
Tunneling includes this entire process (**Encapsulation**, **Transmission**, and **Decapsulation** of packets).

Point-to-Point Tunneling Protocol (PPTP)

GRE – Generic Routing Encapsulation
PPP – Point to Point Protocol



Layer 2 Tunneling Protocol (L2TP)



IP header

UDP header

L2TP header

PPP header

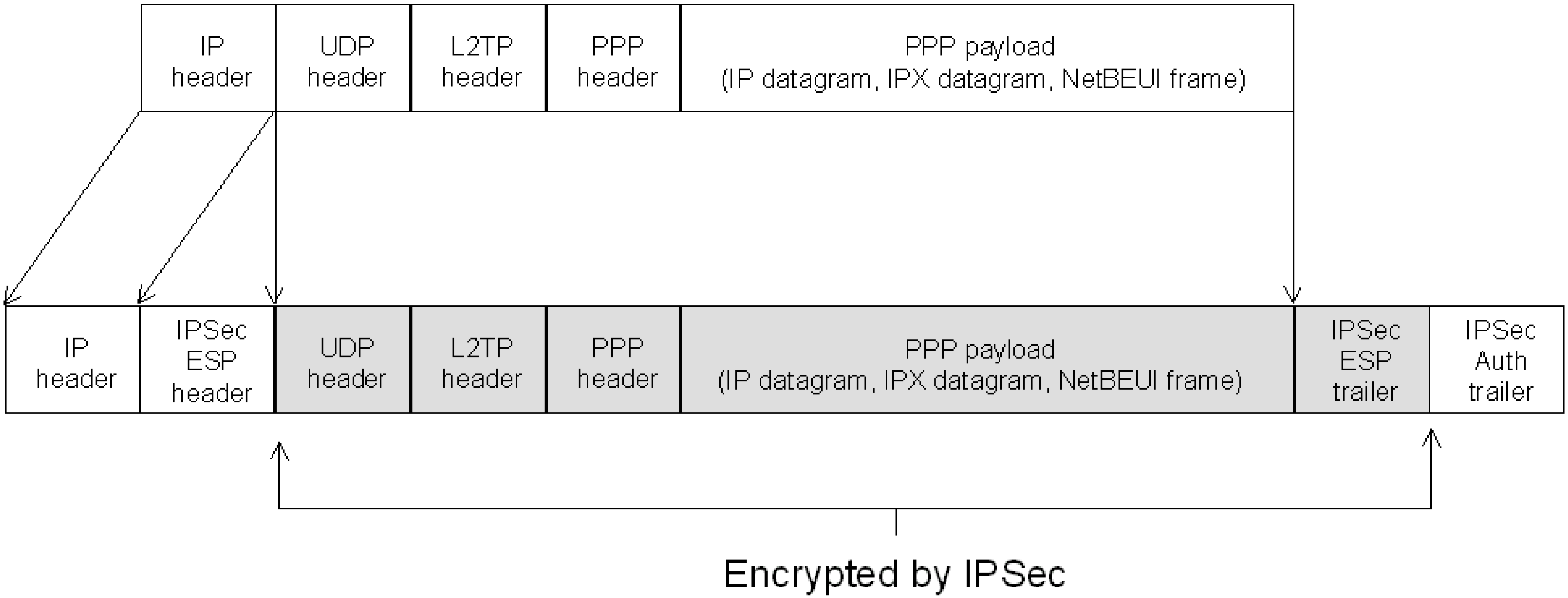
PPP payload (IP datagram, IPX datagram, NetBEUI frame)

PPP frame

L2TP frame

UDP message

Internet Protocol Security (IPSec)

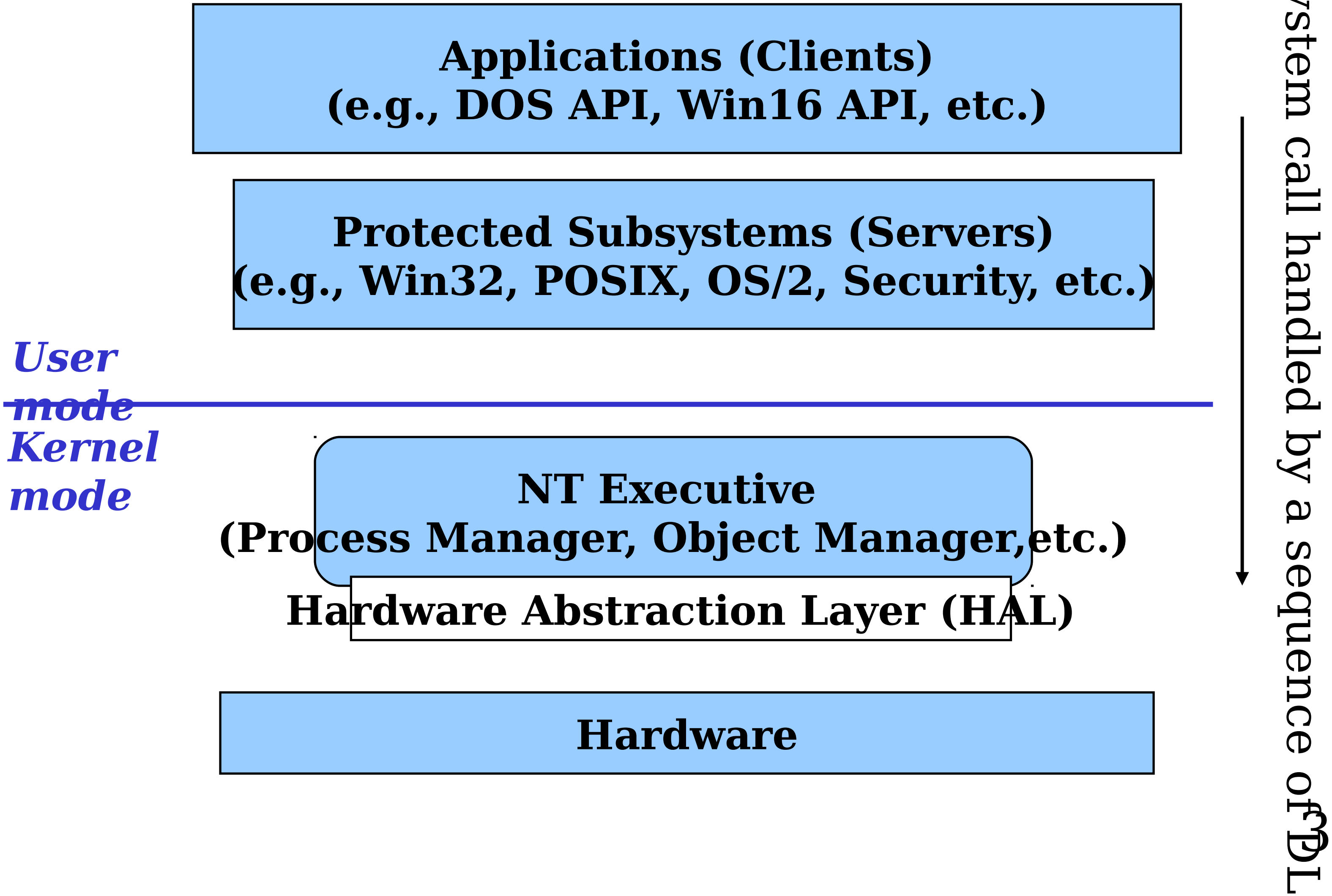


History of Windows 2000/NT

- **1985: Collaborative effort between Microsoft and IBM to build a true multitasking OS, not based on MS-DOS**
  - OS/2
- **1989: Microsoft to build NT, with the following goals**
  - Hardware independent
  - Support of multiple processors
  - Integrated networking (client/server computing) capability
  - POSIX Compliant
  - C2[4] government security certification: *Security Reference Monitor*
- **1993: Windows NT 3.1**
- **1994: Windows NT 3.5**
- **1997: Windows NT 4.0**
  - Borrowed Windows 95 GUI
- **2000-2002: Windows 2000 Professional, Windows XP**

Overview of NT Architecture

- **Micro-Kernel Approach**



Applications (Clients)
(e.g., DOS API, Win16 API, etc.)

Protected Subsystems (Servers)
(e.g., Win32, POSIX, OS/2, Security, etc.)

User mode

Kernel mode

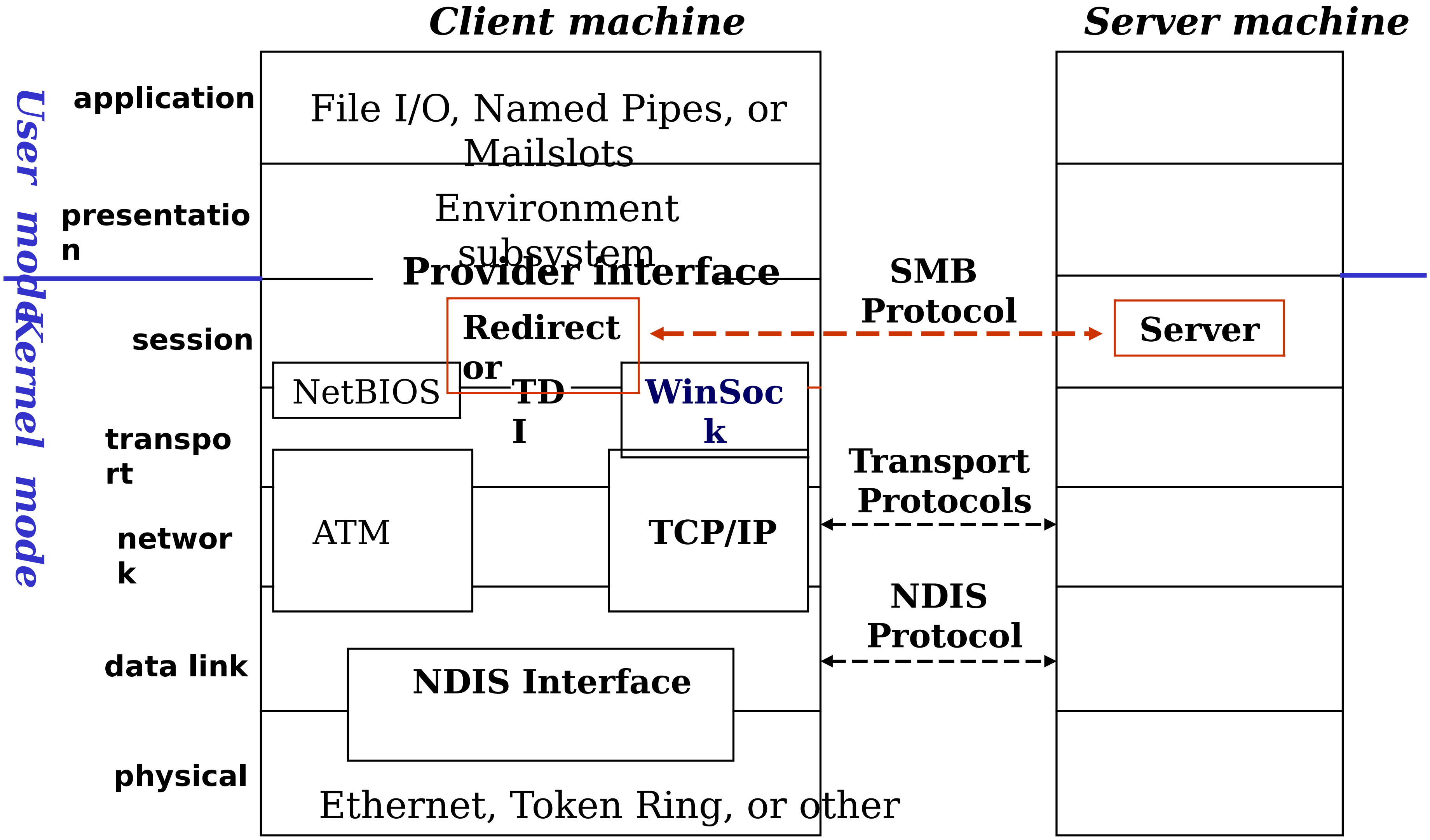
NT Executive
(Process Manager, Object Manager,etc.)

Hardware Abstraction Layer (HAL)

Hardware

A system call handled by a sequence of DL

NT Networking Components



NT Terminology

- **Network Driver Interface Specification (NDIS)**
- **Transport Driver Interface (TDI)**
  - Standard interface for a transport driver to export
- **Redirector**
  - used to locate and set up connection with server when a local logical device is mapped to a network resource by a server
- **Server Message Block (SMB) Protocol**
  - used for communication between redirector and server

NT Redirector

- **A layer at client on top of transport drivers**
  - to support multiple transport protocols

- **Use SMB protocol to communicate with server**
  - to agree on a particular transport protocol (driver)
    - try one transport driver at a time until a server responds
  - File IO example (DOS commands)

```
net  use  h:  \\birdie\SharedSAAM
// map h drive to a share named “SharedSAAM”     // of
remote disk at server named “birdie”
// redirector is called at this point
copy  config.sys  h:\config.sys
// copy config.sys to that drive
net  use   h:   /d           // shut down the connection
```

Communication Methods in NT

- **Pipes**
- **Mailslots**
- **Windows Sockets (WinSock)**
- **NetBIOS** (Network Basic Input/Output System)
  - Interface for NetBIOS Extended User Interface (NetBEUI) protocol
- **SPX/IPX**
  - Novell (Netware) Networks

Pipe

- ***Application-level* construct**
  - bi-directional, connection-oriented
  - transport protocol independent
    - redirector (SMB protocol) is used
  - message-based read/write

- **<u>Named pipe</u> for arbitrary IPC**
  - processes may be on different machines
  - name format: \\<computer name>**PIPE**\<pipe name>

    <computer name> = “.” for local computer

- **<u>Anonymous pipe</u> for IPC on the same machine**
  - between child and parent processes
  - *pipe handle being unknown to non-relative processes*

Mailslot

- ***Application-level* construct**
  - best-effort, one-way communication channel
  - connectionless; use of *broadcast* datagrams
  - transport protocol independent (using redirector services)
  - message-based read/write

- **Name format**
  - when created by a server

    \\.**\mailslot**\<name>
  - when accessed by a client
    - server is local   \\.**\mailslot**\<name>
    - remote   \\<computer name>**\mailslot**\<name>
    - domain   \\<domain name>**\mailslot**\<name>
    - all   \\***mailslot**\<name>

WinSock

- **Kernel level construct**
  - Berkeley-style socket interface (mostly over TCP/IP) in Windows
  - standard API interface for lower layer (transport) protocol
    - bypass redirector
    - transport protocol dependent (specific header files, libraries, etc.)
  - byte-based read/write

- **Name resolution**
  - using DNS
  - using WINS
    - mapping between NetBIOS names and IP addresses
  - using LMHOSTS file

Socket API

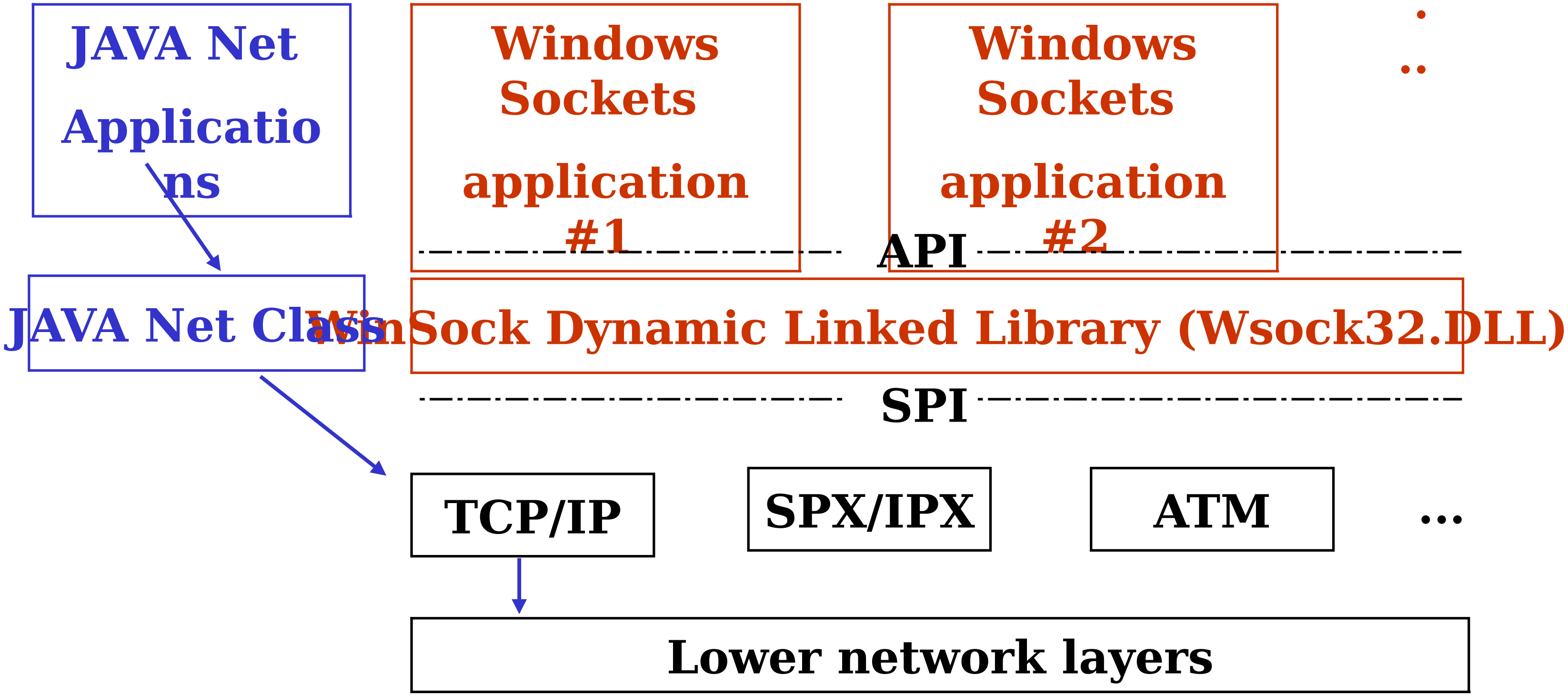
- **“Programmable plug” to the network**
  - standard network interface for applications to build end points of communication channels
- **History**
  - Berkeley Sockets, 1982
  - Socket Interface for TCP/IP on BSD (Unix), 1986
  - Windows Sockets (WinSock 1.1), 1991
    - portable from UNIX at source code level
  - WinSock 2.0, 1995
    - fully backward compatible with WinSock 1.1

WinSock Network Model

- **WinSock application**
  - provides upper-layer functionality (OSI layers 5-7)
- **Network system**
  - provides lower-layer functionality (OSI layers 1-4)
- **WinSock API**
  - allows upper layers access to lower-layer services

WinSock Operation Modes

- **Blocking**
  - “wait on hold until the persons come to the phone”
  - simplest logic; but slowest progress for program
  - solution: multi-threaded programming
- **Nonblocking (polling)**
  - “hang up and call back later” (explicit; or using select( ) system call)
  - fastest “program progress”; however polling incurs a lot of overhead
- **Asynchronous**
  - “leave a message to have the person call you back”
  - OS takes care of message passing; WAsyncSelect( ) of MS Windows

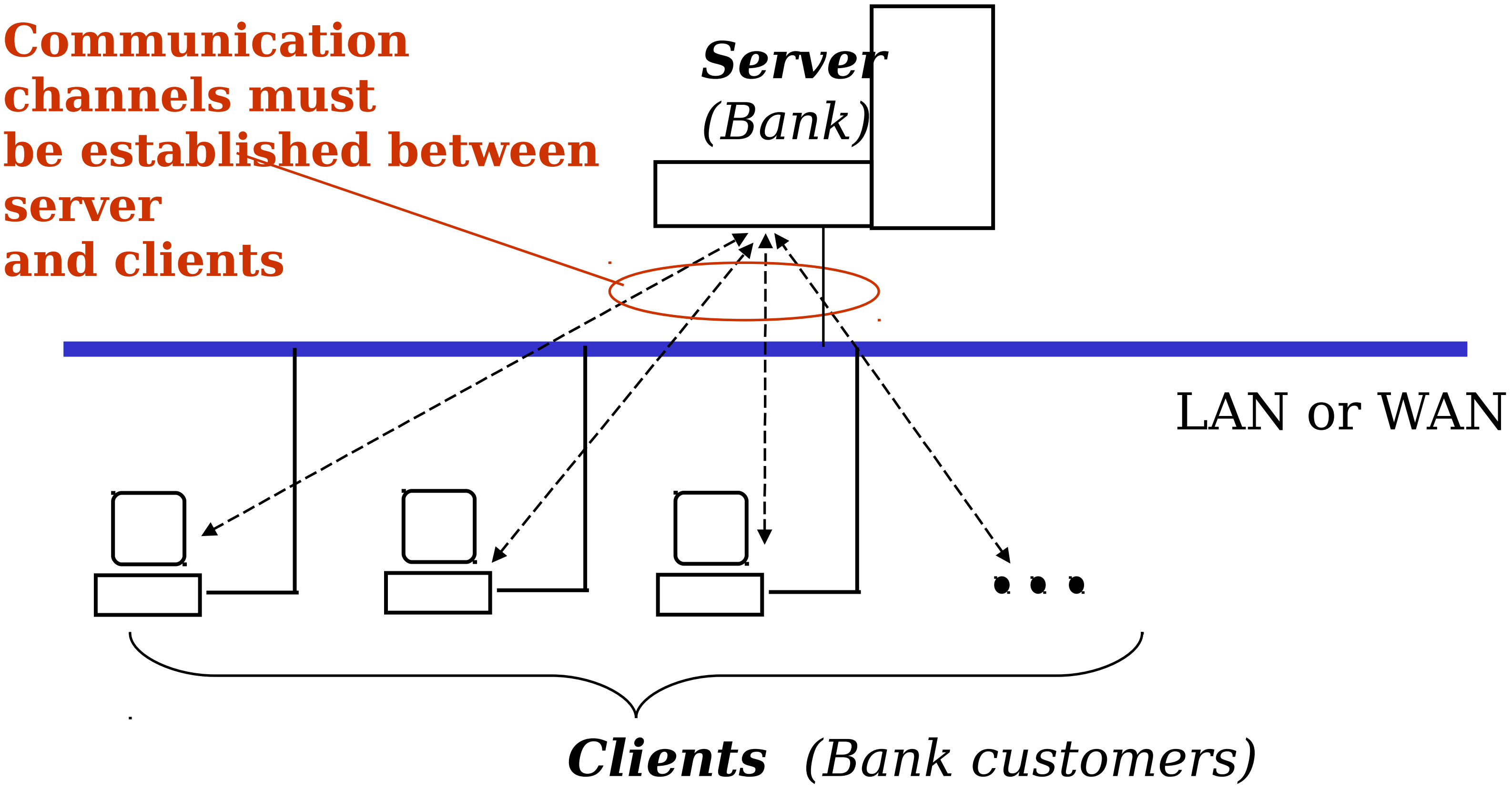

JAVA Net Applicatio ns
Windows Sockets application #1
Windows Sockets application #2
...
API
JAVA Net Class
WinSock Dynamic Linked Library (Wsock32.DLL)
SPI
TCP/IP
SPX/IPX
ATM
...
Lower network layers

- WinSock API is protocol independent
- There are other APIs (e.g., JAVA Socket) for TCP/IP protoco

Socket Programming Specifics

- **C (MS Windows)**
  - initialization/cleanup required: WSAStartup( ) & WSACleanup( )
  - ws2_32.lib (interface to WinSock2 DLL) required
  - debugging tool: WSAGetLastError() (retrieve error code(s))
- **C++ (MS Windows)**
  - CSocket and CAsyncSocket class to define sockets
  - CArchive object to pass data
  - callback functions: OnReceive( ), etc. for Asynchronous mode
- **JAVA (any platform)**
  - JAVA API for TCP/IP implemented in net class

Client/Server Model


Communication channels must be established between server and clients
Server
(Bank)
LAN or WAN
. . .
Clients *(Bank customers)*

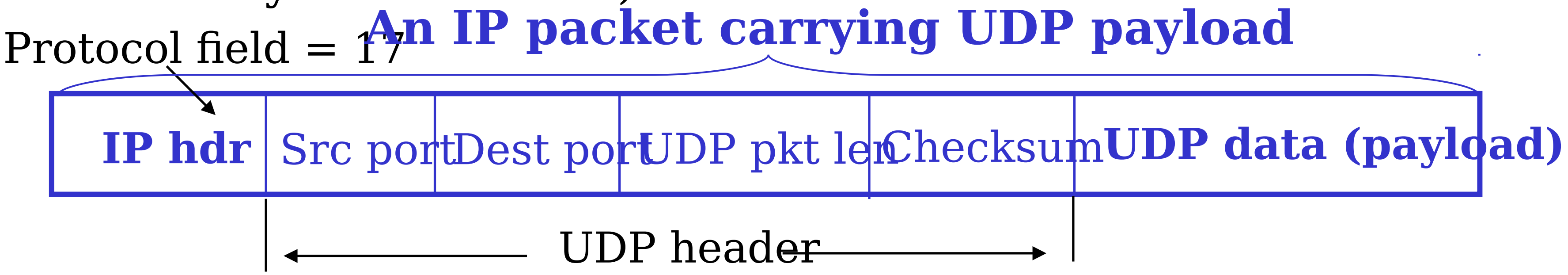
- A client <u>initiates</u> communication with a server

TCP/IP Protocol Suite

- **Application layer protocols**
  - FTP, HTTP, SMTP, DNS, Telnet, etc
- **Transport layer protocols**
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)
- **Network Layer**
  - IP
  - Address Resolution Protocol (ARP)
  - Internet Control Message Protocol (ICMP)

UDP

- **Connectionless transport**
  - unreliable <u>datagram</u> service
    - like regular mail service by Post Office
    - applications may have to do error control themselves
  - low cost
- **Service Access Point**
  - 16-bit port number (1 – 1023 reserved for system services)



An IP packet carrying UDP payload

Protocol field = 17

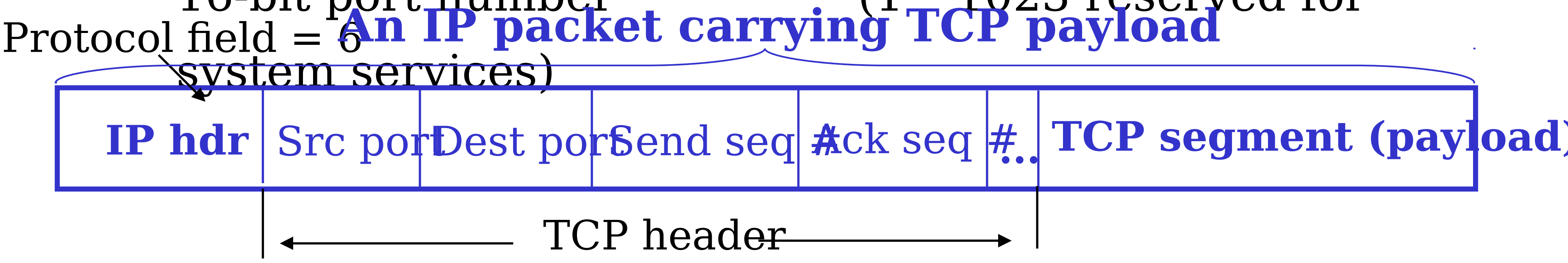
IP hdr | Src port | Dest port | UDP pkt len | Checksum | **UDP data (payload)**

UDP header

TCP

- **Connection-oriented transport**
  - reliable and in-order delivery -- like a stream
    - explicit acknowledgements required from receiver
    - sliding window based flow control
  - high cost because of overhead associated with

    connection management
- **Service Access Point**
  - 16-bit port number (1 – 1023 reserved for system services)



UDP Application Mechanics

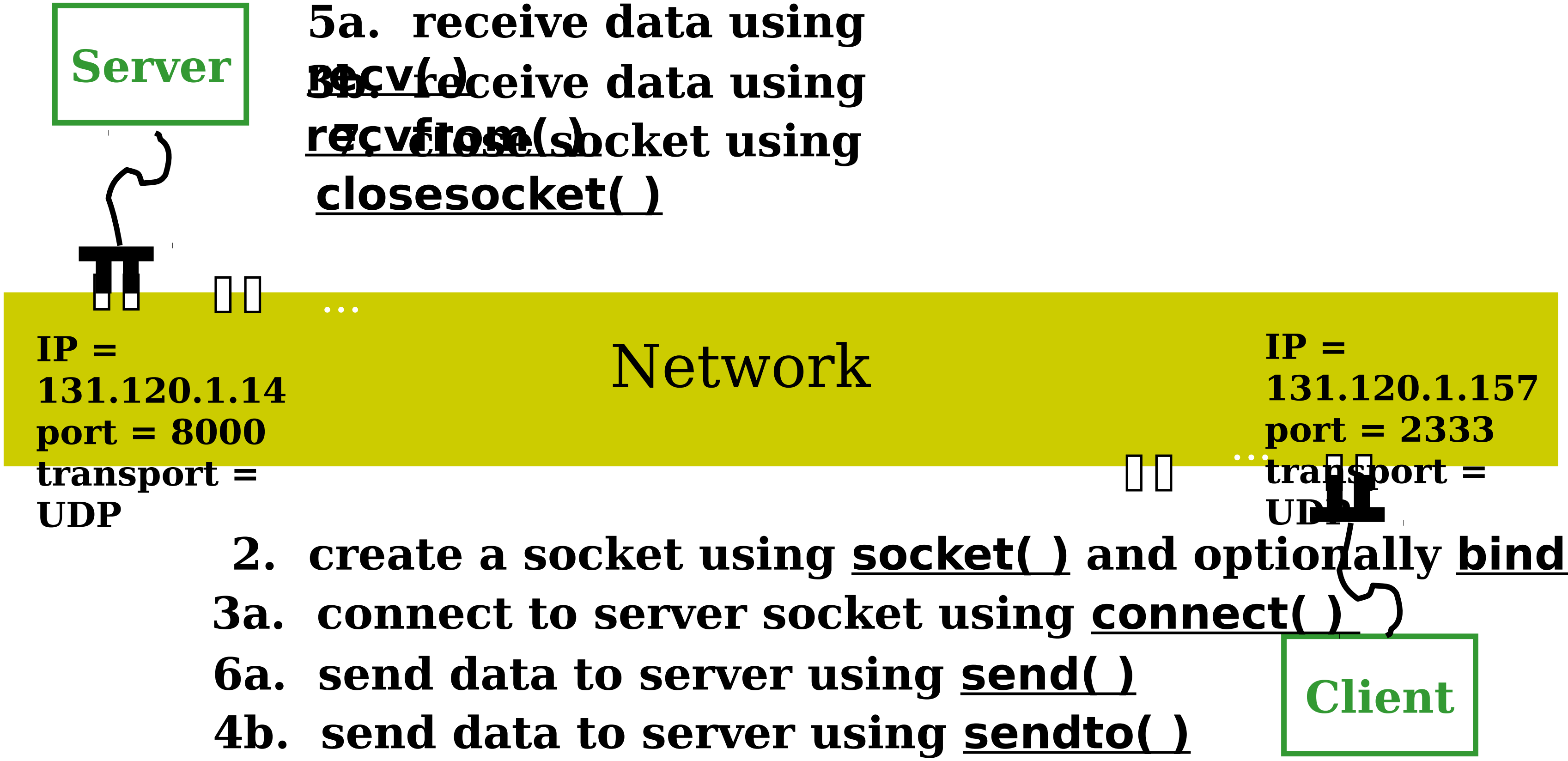
1. create a socket using socket( ) and bind( )

4a. establish socket association using accept( )

5a. receive data using recv( )

3b. receive data using recvfrom( )

7. close socket using closesocket( )



Server

IP = 131.120.1.14
port = 8000
transport = UDP

Network

IP = 131.120.1.157
port = 2333
transport = UDP

Client

2. create a socket using socket( ) and optionally bind( )

3a. connect to server socket using connect( )

6a. send data to server using send( )

4b. send data to server using sendto( )

7. close socket using closesocket( )

TCP Application Mechanics

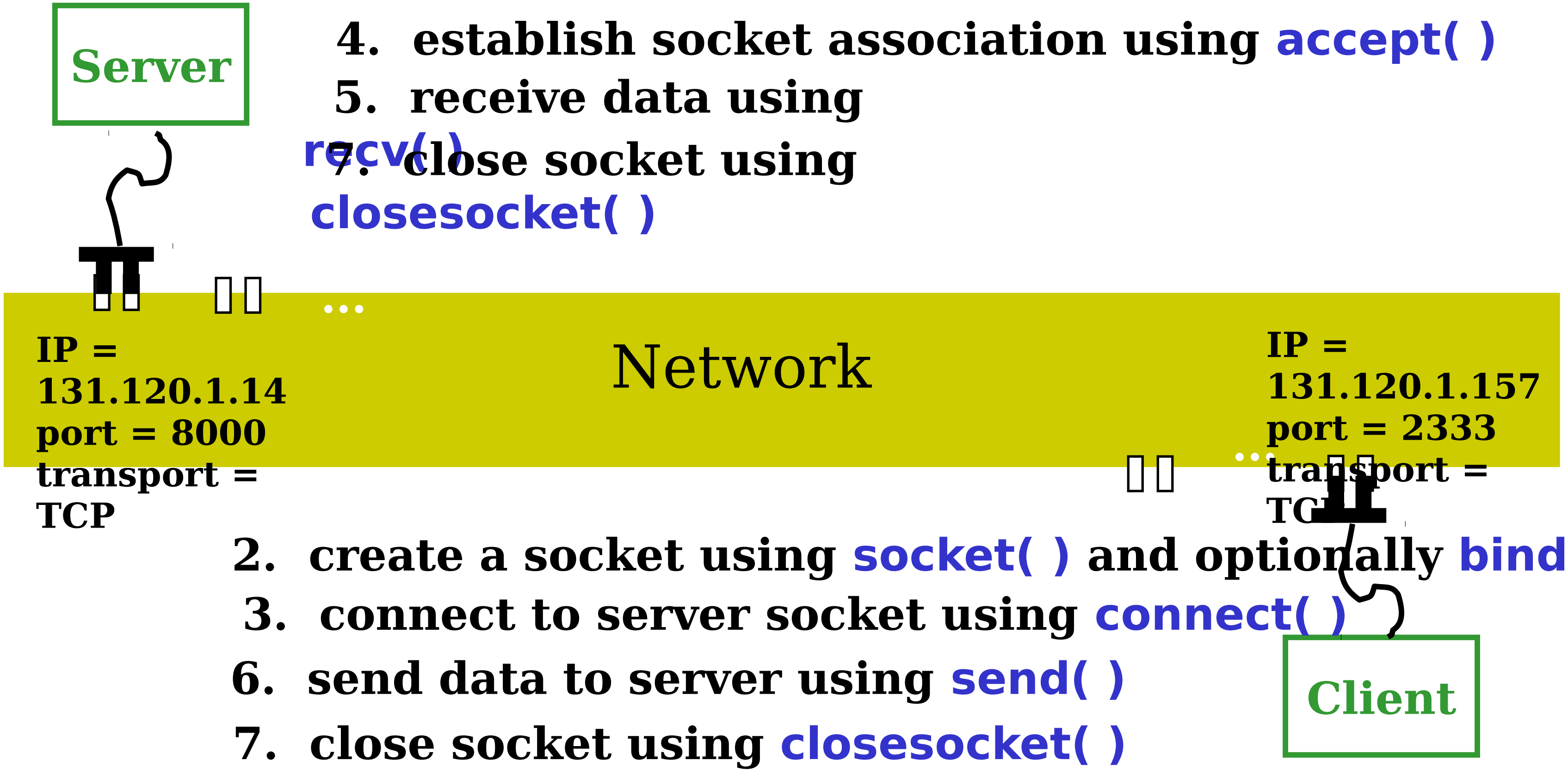
1. **create a socket using socket( ) and bind( )**

2.5. **wait for incoming connections using listen( )**

4. **establish socket association using accept( )**

5. **receive data using recv( )**

7. **close socket using closesocket( )**



Server

Network

IP = 131.120.1.14
port = 8000
transport = TCP

IP = 131.120.1.157
port = 2333
transport = TCP

Client

2. **create a socket using socket( ) and optionally bind( )**
3. **connect to server socket using connect( )**
6. **send data to server using send( )**
7. **close socket using closesocket( )**

C Code Examples

- **Download instruction from a CS machine**
  - ftp to xiepc
    username: “anonymous” and password: <ur email addr>
  - type “cd outgoing” at FTP prompt
  - type “get c-code-examples.zip”
- **Content of c-code-examples.zip**
  - Echo-C: C code for an echo application made of a echo-server and echo-client; UDP sockets are used
  - NonBlocking-C: C code for a modified version of the echo server; server socket is made nonblocking for recvfrom( )

Java Code Example

- **Download instruction from a CS machine**
  - ftp to xiepc:
    username: “anonymous” and password: <ur email addr>
  - type “cd outgoing” at FTP prompt
  - type “get java-code-example.zip”
- **Content of java-code-example.zip**
  - <u>Fortune Cookie</u>: code for server and client; and a flat file for storing “wise” phrases

Java Network Programming

- **Focuses**
  - multi-thread
  - TCP/UDP transport
- **More examples**
  - design and implementation of two SAAM modules
    - routing
    - emulation